INTELLIGENT INSTRUMENTATION®

INNOVATIVE PC Data Acquisition Solutions

UDAS 1001-E Series
U-Link Software Libraries
User Manual
for the
Multifunction Data Acquisition System

A Burr-Brown Company

855M496 1.0

Copyright 2000 by Intelligent Instrumentation Inc., Tucson, Arizona, USA

All rights reserved.

Warranty and Repair Policy Statement

General

Seller warrants that its products furnished hereunder will, at the time of delivery, be free from defects in material and workmanship and will conform to Seller's published specifications applicable at the time of sale. Seller's obligation or liability to Buyer for products which do not conform to above stated warranty shall be limited to Seller, at Seller's sole discretion, either repairing the product, replacing the product with a like or similar product, or refunding the purchase price of the nonconforming product, provided that written notice of said nonconformance is received by Seller within the time periods set forth below:

a. For all software products, including licensed programs, ninety (90) days from date of initial delivery to Buyer;

b. For all hardware products, including complete systems, one year from date of initial delivery to Buyer, subject to the additional conditions of paragraphs c) and d) below;

c. In the event that Buyer's returned product is a Discontinued product and is irreparable for any reason, Seller may elect to replace it with like or similar product that is, in Seller's sole judgment, the closest equivalent to the returned product. Seller does not warrant that such replacement product will be an exact functional replacement of the returned product.

Further, all products warranted hereunder for which Seller has received timely notice of nonconformance must be returned FOB Seller's plant no later than thirty (30) days after the expiration of the warranty periods set forth above.

These warranties provided herein shall not apply to any products which Seller determines have been subjected, by Buyer or others, to operating and/or environmental conditions in excess of the limits established in Seller's published specifications or otherwise have been the subject of mishandling, misuse, neglect, improper testing, repair, alteration or damage. THESE WARRANTIES EXTEND TO BUYER ONLY AND NOT TO BUYER'S CUSTOMERS OR USERS OF BUYER'S PRODUCT AND ARE IN LIEU OF ALL OTHER WARRANTIES WHETHER EXPRESS, IMPLIED OR STATUTORY INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SELLER BE LIABLE FOR INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES. Seller's liability for any claim of any kind shall in no case exceed the obligation or liability specified in this Warranty clause.

Technical Assistance and Service

Seller's warranty as herein set forth shall not be enlarged, diminished or affected by, and no obligation or liability shall arise or grow out of, Seller's rendering of technical advice, facilities or service in connection with Buyer's order of the goods furnished hereunder. Products returned for warranty service, but which are found to be fully functional and in conformance with specifications may be subject to a nominal service charge and return freight charges. Periodic re-calibration of products, if required, is the responsibility of Buyer and is not provided under this Warranty.

Online Support

If errors persist after repeatedly checking the unit as described in the *UDAS-1001E Series Hardware User Manual*, check the technical support links on Intelligent Instrumentation's 24-hour World Wide Web site at **http:// www.instrument.com**. The site contains information on Intelligent Instrumentation's products, new developments, announcements, application notes, application examples, and other useful information. The site and support areas continue to grow as new products, updates, and features are added.

Email Support

Intelligent Instrumentation's technical support can be reached via email. When sending an email message, be sure to include complete contact information as well as a detailed description of the products being used and the problem. Use the following addresses to contact the support group:

North American Customers

us_support@instrument.com

app_eng@instrument.com

International Customers

international@instrument.com

int_support@instrument.com

In France:france@instrument.com

In Italy: italy@instrument.com

In Germany: germany@instrument.com

Static Sensitivity

Seller ships all static-susceptible products in anti-static packages. Seller's Warranty as herein set forth shall not cover warranty repair or replacement for products damaged by static due to Buyer's failure to use proper protective procedures when handling, storing, or installing products.

Trademarks

U-Link ™Software Libraries is a trademark of Intelligent Instrumentation Incorporated.

VisualDesigner® is a registered trademark of Intelligent Instrumentation Incorporated.

LabVIEW® is a registered trademark of National Instruments Corporation.

Windows® 98 is a registered trademark of Microsoft® Corporation in the U.S.A. and other countries.

Other products or brand names are trademarks or registered trademarks of their respective companies.

Use of Equipment

Intelligent Instrumentation Inc., assumes no responsibility for any direct, indirect or consequential loss or damages resulting from misuse of the equipment or for improper or inadequate maintenance of the equipment or for any such damage or loss resulting from the use of other equipment, attachments, accessories, and repairs at any time made to or placed upon the equipment or any replacement thereof. Furthermore, Intelligent Instrumentation Inc., makes no representations or warranties, either expressed or implied, in connection with the use of the equipment in the event it is improperly used, repaired or maintained.

FCC Radio Frequency Interference Statement

This equipment generates and uses radio frequency energy, and may cause interference to radio or television reception.

Per FCC rules, Part 15, Subpart J, operation of this equipment is subject to the conditions that no harmful interference is caused and that interference must be accepted that may be caused by other incidental or restricted radiation devices, industrial, scientific or medical equipment, or from any authorized radio user.

The operator of a computing device may be required to stop operating his device upon a finding that the device is causing harmful interference and it is in the public interest to stop operation until the interference problem has been corrected.

The user of this equipment is responsible for any interference to radio or television reception caused by the equipment. It is the responsibility of the user to correct such interference.

UDAS-1001E Series User Manual
Revision History

Version	Date	Revision
1.0	000217	Original Release

Table of Contents

Chapter 1 U-Link Software Libraries: Introduction and Installation

1.1 About this Manual 1-1
1.2 UDAS-1001E-1 Series U-Link Software Libraries 1-1
1.3 Introduction 1-2
1.3.1 System Configuration Requirements 1-3
1.3.1.1 Microsoft Windows Environment Programming Language Support 1-3
1.3.2 The Software License Agreement 1-3
1.4 Software Installation 1-3

Chapter 2 Programming the UDAS Unit

2.1 Application Programming Interface 2-1
2.2 Initialization and De-initialization Functions 2-2
2.2.1 Programming Procedures 2-2
2.3 Digital I/O Port Functions 2-3
2.3.1 Programming Procedures 2-4
FIGURE 2.1 Reading and Writing Digital I/O Port Data 2-5
2.4 Digital I/O Bit Functions 2-6
2.4.1 Programming Procedures 2-8
2.4.1.1 Programming Digital Input Bit Modes 2-8
FIGURE 2.2 Digital Input Bit Mode 2-10
FIGURE 2.3 Digital Trigger Input Mode 2-11
2.4.2 Programming Digital Output Bit Modes 2-12
FIGURE 2.4 Digital Output Bit Mode 2-13
2.5 Counter Function Calls 2-14
FIGURE 2.5 Event Counter Block Diagram 2-15
FIGURE 2.6 Period Measurements Block Diagram 2-16
FIGURE 2.7 Frequency Measurements 2-17
2.6 Analog Input Function Calls 2-18
FIGURE 2.8 Software Controlled Analog Input Read 2-21
2.7 High-Speed Analog Input Functions 2-22
2.7.1 Pacers and Triggers 2-23
FIGURE 2.9 High-Speed Analog Input Acquisition 2-29
2.8 Analog Output Function Calls 2-31
2.9 Analog Output High-Speed Functions 2-32
FIGURE 2.10 Writing High-Speed Analog Output Data 2-35
2.10 Rate Generator Function Calls 2-36
FIGURE 2.11 Rate Generator Programming 2-38
2.11 Trigger Function Calls 2-39
2.12 Miscellaneous Functions 2-40
2.12.1 uSetOption Function Calls 2-40
2.12.2 Utility Function Calls 2-41

Appendix A U-Link Software Libraries Function Call Reference

A.1 Introduction A-1
A.2 Run-Time Library Reference A-1
A.3 Function Calls A-2
TABLE A.1 Function Calls A-2
A.4 Function Call Specifications A-4
A.4.1 Presentation of Call Specifications A-4
A.4.2 Supported Software Libraries A-4
A.4.3 Error Codes A-6
TABLE A.2 System Error Codes A-6
A.5 Initialization Functions A-7
A.5.1 uDeviceOpen A-7
A.5.2 uDeviceClose A-8
A.5.3 uSYSInquire A-8
A.6 Analog Input Functions A-10
A.6.1 uAIMAXEnable A-10
A.6.2 uAIConfigureList A-11
A.6.3 uAIRead A-13
A.7 Analog Input High-Speed Functions A-16
A.7.1 uAIHSConfigureList A-16
A.7.2 uAIHSEnable A-18
A.7.3 uAIHSRead A-20
A.7.4 uAIHSReturn A-21
A.7.5 uAIHSStatus A-22
A.8 Analog Output Functions A-23
A.8.1 uAOConfigureList A-23
A.8.2 uAOWrite A-24
A.9 Analog Output High-Speed Functions A-25
A.9.1 uAOHSConfigureList A-25
A.9.2 uAOHSEnable A-26
A.9.3 uAOHSStatus A-27
A.9.4 uAOHSWrite A-28
A.10 Counter Functions A-29
A.10.1 uCTRConfigure A-29
A.10.2 uCTREnable A-31
A.10.3 uCTRRead A-32
A.10.4 uCTRStatus A-33
A.11 Digital Input/Output (Port) Functions A-34
A.11.1 uDIOConfigureList A-34
A.11.2 uDIORead A-35
A.11.3 uDIOWrite A-36

A.12 Digital Input/Output (Bit) Functions A-36
A.12.1 uDIOBITConfigureList A-36
A.12.2 uDIOBITRead A-39
A.12.3 uDIOBITWrite A-40
A.13 Rate Generator Functions A-41
A.13.1 uRGConfigure A-41
A.13.2 uRGEnable A-42
A.14 I/O Option Function A-44
A.14.1 uSetOption A-44
A.15 Trigger Function A-45
A.15.1 uTRIGConfigure A-45
A.16 Utility Functions A-46
A.16.1 uCountsToTemperature A-46
A.16.2 uCountsToVolts A-48
A.16.3 uFrequencyToRGCounts A-48
A.16.4 uVoltsToCounts A-49

Appendix B Compiling, Linking, and Running U-Link Applications
B.1 General Information B-1
B.2 Compiling and Linking B-1
B.3 Running B-2

Appendix C Error Code Information
C.1 Successful Operation Code C-1
C.2 Error Codes Returned by UDAS Driver DLL C-1
C.3 Error Codes Returned by the UDAS Driver DLL C-3
C.4 Multi-function Error Codes C-4
C.5 Digital I/O Error Codes C-4
C.6 Analog Input Error Codes C-5
C.7 Analog Output Error Codes C-6
C.8 Rate Generator Error Codes C-6
C.9 Trigger Error Codes C-7
C.10 High-speed Analog Input Error Codes C-7
C.11 High-speed Analog Output Error Codes C-9
C.12 Counter Error Codes C-10
C.13 Thermocouple Error Codes C-11
C.14 Miscellaneous Error Codes C-12

Chapter 1 U-Link Software Libraries: Introduction and Installation

1.1 About this Manual

This manual is a guide to using all versions of the U-Link Software Libraries UDAS Multifunction Data Acquisition System (UDAS) units and is organized into chapters describing how to install and use the functions provided in U-Link. Appendix A, *U-Link Software Libraries Function Call Reference* provides a complete listing of all function calls including call descriptions, syntax, parameter listings and other necessary information. Appendix B, *Compiling and Linking* provides compiling and linking instructions for the supported operating environments.

The contents of this manual includes:

Chapter 1 Provides an introduction to this manual and a brief overview.

Chapter 2 Instructions for installing the U-Link Software Libraries.

Chapter 3 Describes U-Link initialization functions.

Appendix A Describes the U-Link Software Libraries in detail. Includes a complete description of each software call.

Appendix B Provides compiling and linking instructions for each supported environment.

Appendix C Provides error information.

1.2 UDAS-1001E-1 Series U-Link Software Libraries

The UDAS-1001E-1 Series U-Link Software Libraries U-Link provides an easy means of programming any UDAS unit.

The Libraries Include function calls for:

- Configuring each of an UDAS unit's I/O functions.
- Calls for reading and writing analog I/O and digital I/O data.
- Various utility functions for simplifying routine operations.

In addition to Microsoft Visual C and Microsoft Visual Basic, the most popular C/C++ compilers are supported by the software libraries. All supported compilers are listed in section 1.3.1.1, *Microsoft Windows Environment Programming Language Support*, and Appendix B, *Compiling, Linking, and Running U-Link Applications*.

U-Link software simplifies the interface between the programmer and the hardware. There are no sequences of bit-oriented, register-specific set up calls and no detailed bus procedures. All library function calls return an integer result or error code, indicating the success or failure of the function.

1.3 Introduction

The UDAS-1001E Data Acquisition System CD contains all of the Software Libraries and related files for communicating with applications created with the supported compilers, as well as a variety of sample programs.

To use the Libraries, install the libraries and make the connections between the DLL files and your application program by calling the U-Link API functions described in this manual. The application programming interface is provided through functions included in the file **I3USB.DLL**. These functions and associated structures are defined in the header file **USBFCNS.H**.

An additional header file, **USBDEFS.H**, provides definitions of constants that are specified as options for some the functions. Borland projects must include **BORIUSB.LIB** in order to access the **I3USB.DLL**. Microsoft projects must include **MICIUSB.LIB** in order to access the **I3USB.DLL**. The header file **I3USBERR.H** contains the presently defined error codes.

Appendix B, *Compiling and Linking*, contains important information on compiling and linking your programs. Review the **README.TXT** files supplied with the software; they may contain new information not included in this manual.

Chapter 2 is divided into the following topics:

- System configuration requirements
- Program license agreement
- Installing the U-Link Software Library files for each supported operating environment

1.3.1 System Configuration Requirements

The personal computer system that you use can be an IBM compatible PC (486 or later). The Microsoft Windows 98 operating environment is supported.

1.3.1.1 Microsoft Windows Environment Programming Language Support

To write programs using the U-Link for Windows Software Libraries, use one of the supported compilers listed below:

- Borland C++
- Microsoft Visual Basic
- Microsoft Visual C++
- other DLL compatible C compilers

1.3.2 The Software License Agreement

Read the Program License Agreement printed on the envelope containing the UDAS-1001E Data Acquisition System CD. If you have any questions, contact your sales representative before continuing.

1.4 Software Installation

U-Link software installation is accomplished by running the **SETUP.EXE** program on the UDAS-1001E Data Acquisition System CD and following the instructions.

Chapter 2 Programming the UDAS Unit

2.1 Application Programming Interface

The application programming interface is implemented within the Dynamic Link Library **I3USB.DLL**. These functions and associated structures are defined in the header file **USBFCNS.H**. An additional header file, **USBDEFS.H**, provides definitions of constants that are specified as options for some of the functions. In order to access the API functions, the import library (**MICIUSB.LIB** for MS Visual C++ or **BORIUSB.LIB** for Borland C++) must be linked with your application.

This chapter includes descriptions of Application Programming Interface functions primarily intended for application development. Each section discusses a different function, and the sections present the information with the following sub-sections:

Purpose: Briefly explains the reasons for using the functions.

Functions: Lists the functions included in each section.

Programming Procedures: Explains how to organize the functions in a program.

The functions included in this section are defined for the host (PC) application. Appendix A, *U-Link Software Libraries Function Call Reference*, contains detailed information on each function including: the function, parameters, call description, system information structure, and returned error codes.

> **Note:** All of the functions have a zero return value if they successfully complete, otherwise they return an error or warning code.

> **Note:** The U-Link drivers have single-threaded capability. When writing an application in which multiple threads access the drivers, ensure that a thread is not interrupted during an execution of a U-Link function.

2.2 Initialization and De-initialization Functions

Purpose

The descriptions of the functions in this section are used to:

- Begin and end communication sessions with UDAS units.
- Obtain system information about a connected UDAS unit.

Functions

uDeviceOpen opens the UDAS device and performs an initial test.
uDeviceClose closes the handle to the UDAS device.
uSYSInquire returns information about the connected UDAS device.

2.2.1 Programming Procedures

When creating your program, follow the organization of the initialization function calls that are summarized below. The software package includes sample programs which provide a demonstration of these function calls.

1. Write the initialization sequence for each UDAS device being used in your application. The first initialization function call to use in your program is **uDeviceOpen**, a mandatory function that must precede *all* other U-Link Software Library calls in the program. The **uDeviceOpen** U-Link Driver function allocates system resources required by the drivers.

   uDeviceOpen does the following:
   - Opens the UDAS device.
   - Performs an initial test.
   - Sends reset and inquiry commands to the UDAS unit and stores the SYSInquire command. It also stores the data that can be passed to an application.

   > **Note: uDeviceOpen** may be called more than once if there is more than one UDAS unit connected to the host.

   After opening the device, issue the **uSYSInquire** command. The **uSYSInquire** function returns system information about a connected UDAS unit, including its serial number and version number. Although this function is optional, it is extremely useful in most programs.

2. Write the configuration calls that you plan to include in the program.

> **Note:** When using an AI-MAX unit to expand the configuration to include more than 16 channels, the **uAIMAXEnable** function must be called *before* the program calls any analog or digital functions.

3. Write the application for your program.
4. Before exiting your program, call **uDeviceClose** to de-initialize the U-Link software. This is a mandatory function that is *always* the last U-Link call in the program.

   uDeviceClose frees resources and closes any open links with UDAS units.

> **Note:** Call **uDeviceClose** once for *each* device that has been opened.

2.3 Digital I/O Port Functions

Purpose

There are two digital I/O ports of 8 bits each on a UDAS unit. Digital I/O port functions can be used to read or write a group of related signals as a data byte. Digital I/O port functions may also be used to read unrelated signal levels simultaneously, and to read or write the current state of bits which have been configured as bit signals (see section 2.4, *Digital I/O Bit Functions*).

Functions

uDIOConfigureList is the call to use to configure of a list of digital I/O ports on a UDAS unit.
uDIORead reads the digital I/O data from the ports.
uDIOWrite writes data to the digital output ports.

Configuring Input and Output Ports

The port list is specified through two **uAOConfigureList** parameters: *list* and *count*. Either port can be configured independently as an input or an output.

The *list* parameter uses a defined digital I/O port list structure called **uDIOListType**. Use the *count* parameter to tell the function the number of ports to configure (maximum of 2 for a UDAS-1001E). The list that you create is an array of *count* elements, of type **uDIOListType**.

Reading and Writing Digital Data

There are two methods for specifying the port to be read or written:

- Either as a port number, 0 or 1, or
- Use the code **ALL_PORTS**, which accesses port 0 and port 1.

  The data is returned in the buffer pointed to by the call's data parameter.

If all ports or multiple ports are read or written, dimension the data buffer for the number of digital I/O ports on the unit.

2.3.1 Programming Procedures

After the proper system initialization calls (FIGURE 2.1, *Reading and Writing Digital I/O Port Data*) and the required **uDIOConfigureList** call have been made, the **uDIORead** and **uDIOWrite** functions can be called to read data from and write data to digital I/O ports.

The **uDIOConfigureList**, **uDIORead** and **uDIOWrite** call descriptions in Appendix A, *U-Link Software Libraries Function Call Reference*, provide all of the information for using these functions. This includes syntax, functional descriptions, parameter lists and their possible values. Refer to these lists for port codes and other requirements.

A sample procedure for reading and writing port data using U-Link Software Library functions is described below and shown in FIGURE 2.1, *Reading and Writing Digital I/O Port Data.*

Example: Reading and Writing Port Data

1. To create a digital I/O port list, use the **uDIOConfigureList** call.

   Port list elements use the **uDIOListType** structure. Each list element specifies the port number, I/O direction (input or output) and initial output data (if output is selected).
2. To configure the ports in the port list, call **uDIOConfigureList** which specifies the number of ports in the list in the call's *count* parameter.
3. To read digital input port data, call **uDIORead**, specify a specific port to read, 0 or 1, or all ports (**ALL_PORTS**) on the UDAS unit. Pass a pointer to an appropriate sized data buffer.
4. To write digital output port data, call **uDIOWrite**, specify a specific port to write, 0 or 1, or all ports (**ALL_PORTS**) on the UDAS unit. Pass a pointer to a buffer containing the data to write.

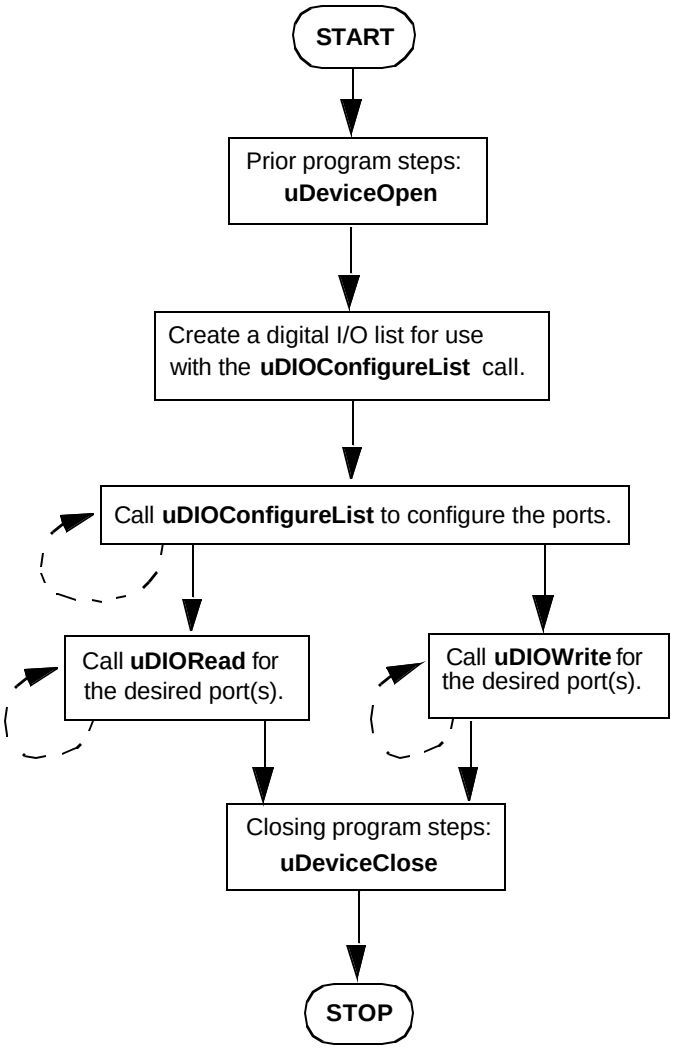

FIGURE 2.1 Reading and Writing Digital I/O Port Data

2.4 Digital I/O Bit Functions

Purpose

To read or write individual unrelated signals separately (not as a group), or to use a digital input signal as counter or trigger, use digital I/O bit functions. The digital I/O port functions may also be used to read or write the signal levels as a group (see section 2.3).

Functions

uDIOBITConfigureList configures a list of digital I/O bits.

uDIOBITRead is a general purpose digital input bit read function.

uDIOBITWrite sets the value(s) for a digital output bit or bits.

Using Trigger Inputs

Digital bits can be configured as trigger inputs. These inputs may cause an interrupt to the UDAS on the rising or falling edge of a signal, or on both the rising and falling edges of a signal. When a specific trigger event occurs, a count is incremented. The number of counts detected is reported when the inputs are read and optionally reset to 0.

Portbit Parameter

The digital bit input and output calls use a *portbit* parameter. If the high byte of this parameter is 0, the low byte contains a bit number (0 through 7). If the high byte is a non-zero value, it contains a mask of bits. The code **ALL_PORT_BITS = 0xFF00** selects all of the bits in the port.

Whenever the portbit parameter is given as a bit mask in a read or write operation, the data is referenced as an 8-element array. For write operations, only the active bits are written. For read operations, data returned for inactive bits are set to 0.

When the portbit parameter is given as a bit number, the data pointer refers to a single data item, not an array.

Configuring the Digital I/O Bits

To configure a list of digital I/O bits on a UDAS unit, use the **uDIOBITConfigureList** call. Digital I/O bits must be configured using this call *before* input or output data can be read from or written to these bits. The initial data value is written to output bits immediately after they are configured as outputs.

Important: Do not mix input and output functions on the same digital I/O port. All configured bits residing on the same port must be either an *input* type or an *output* type. Unused bits do not need to be configured.

Descriptions of Input and Output DIO Modes

DIO_MODE_INPUT

This mode allows direct (unlatched) reading of a digital input bit. A read of a bit configured for normal mode returns the current level (1 or 0) of the input signal.

DIO_MODE_OUTPUT

Writing to a bit configured for **DIO_MODE_OUTPUT** sets the bit's output level 1 or 0.

Note: AI-MAX uses bits 4 through 7 of Port 1. If AI-MAX is used, bits in Port 1 may only be configured as output, and only bits 0-3 may be configured or written.

DIO_MODE_HItoLO and DIO_MODE_LOtoHI

These trigger modes on the UDAS device increment a counter on both high-to-low or low-to-high signal edge.

DIO_MODE_ANY

This trigger mode allows the UDAS device to increment a counter on both high-to-low and low-to-high signal edges.

Reading Digital Input Bit Data

The **uDIOBITRead** call reads a *single* digital I/O bit, *selected* bits (using a portbit mask), or *all* bits on a specified port. The **uDIOBITRead** call also returns a status word which is a bit mask indicating the enable status for trigger inputs.

- For bits configured as inputs, the level of the input bit is returned (1 or 0).
- For bits configured as outputs, the last level written to the bit is read back (1 or 0).
- Bits that are configured as trigger inputs return the current trigger count.
- If the reset parameter is *true*, the trigger counts are set to 0 after the reading occurs.

- The status parameter returns a bit mask using a value of 1 to indicate the bits configured as trigger inputs that are currently armed.
- The *portbit* parameter may specify a bit mask or a bit number. If a bit mask is specified, data is returned in an 8-element array, even if only one bit is specified. Data for bits that are not specified by the bit mask are returned as 0.
- If the *portbit* specifies a bit number, the result is returned as a single data value in the first element of data.

Enabling and Disabling Digital Input Bits

The **uDIOBITWrite** function enables or disables digital input bits configured as trigger inputs. The data value 1 enables the bit and the value 0 disables the bit.

- If the bit is configured as a trigger input, the data value 1 arms the trigger. Data value 0 disarms the trigger.
- The **uDIOBITWrite** function must be called to disable a trigger bit before it can be reconfigured.
- **uDIOBITWrite** is not valid for bits configured as ordinary inputs.

Writing Digital Output Bit Data

The **uDIOBITWrite** function sets the value(s) for a single digital output bit, selected bits (using a portbit mask), or all bits on a specified port.

- The data level 0 or 1 is written to the bit without disturbing other bit values.
- If the *portbit* parameter specifies a mask rather than a bit number, data is passed in an 8-element array.
- If *portbit* specifies a bit number, the single data value is taken from the first element of data.

2.4.1 Programming Procedures

This section includes the digital input bit and output bit programming procedures in an instructional and flowchart format. Information on each of the digital I/O bit functions, including a description, format information, and a parameter list is provided in Appendix A, *U-Link Software Libraries Function Call Reference*.

2.4.1.1 Programming Digital Input Bit Modes

This section provides written procedures on how to configure each digital input bit mode and read the input data for each mode.

Input Mode

A sample procedure for programming normal mode digital input bits using U-Link Software Library functions is described below. See FIGURE 2.2, *Digital Input Bit Mode.*

Example: Programming Normal Mode Digital Input Bits

1. Create a bit list for use by the **uDIOBITConfigureList** call.

   Bit list elements use the **uDIOBITListType** structure. The bit number(s) and mode, **DIO_MODE_INPUT** are specified for each list element.
2. Call **uDIOBITConfigureList** to configure the digital I/O bits specified in the bit list. Specify the number of bits in the list in the call's *count* parameter.
3. To read data from bits configured for input mode, call **uDIOBITRead** for a single bit, all bits on a specified port, or selected bits (using a port bit mask). Specify the port number in the call's port parameter.

The data is returned in the buffer pointed to by the data parameter. When using **ALL_PORT_BITS** or a port bit mask, you must dimension the data buffer for 8 elements. Data for the selected bits is returned in the corresponding data buffer elements. The data value indicates the bit's input level, 1 for high or 0 for low.

Digital Trigger Input Mode

A sample procedure for programming trigger mode digital input bits using U-Link Software Library functions is described below. See FIGURE 2.3, *Digital Trigger Input Mode.*

Example: Programming Trigger Mode Digital Input Bits

1. Create a bit list for use by the **uDIOBITListType** structure. The bit number(s) and mode, **DIO_MODE_HItoLO**, **DIO_MODE_LOtoHI, or DIO_MODE_ANY** are specified for each list element, depending on whether you want the bit to trigger on high-to-low or low-to-high transitions, or both.
2. Call **uDIOBITConfigureList** to configure the digital I/O bits specified in the bit list. Specify the number of bits in the list in the call's *count* parameter.
3. To read data from bits configured for input mode, call **uDIOBITRead** for a single bit, all bits on a specified port, or selected bits (using a port bit mask). Specify the port number in the call's port parameter.

The data is returned in the buffer pointed to by the data parameter. When using **ALL_PORT_BITS** or a port bit mask, you must dimension the data buffer for 8 elements. Data for the selected bits is returned in the corresponding data buffer elements. The data value indicates the bit's input level, 1 for high or 0 for low.

The data value indicates the number of triggers that have occurred on that bit since the last time the bit was enabled or reset. The reset parameter indicates whether the count value should be reset after the read (nonzero) or allowed to accumulate (zero). For the current state (high or low) of a trigger input bit, you must read the port using **uDIORead** and extract the level for that bit from the byte value returned.

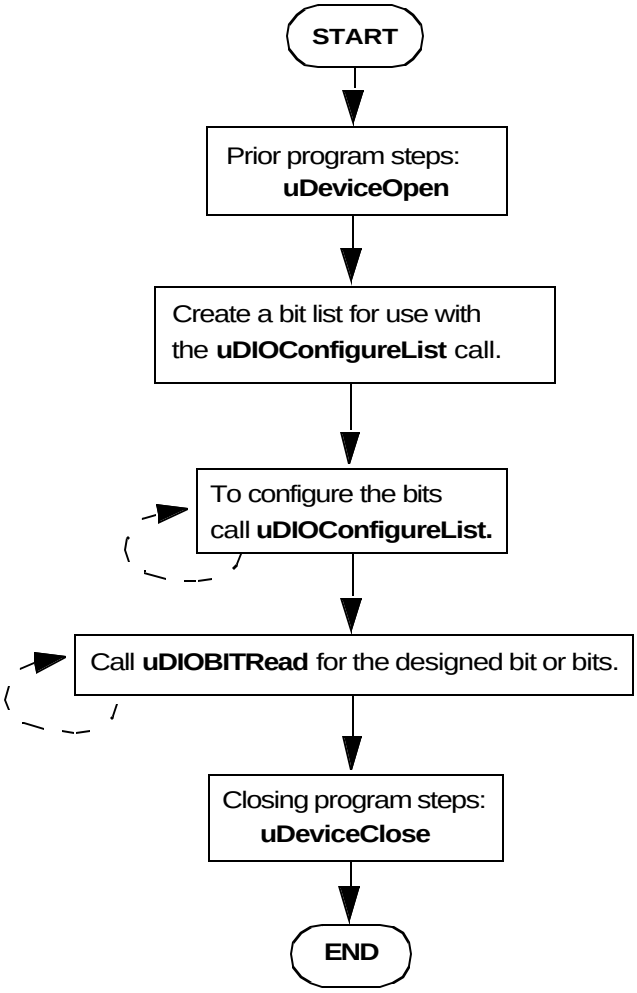

FIGURE 2.2 Digital Input Bit Mode

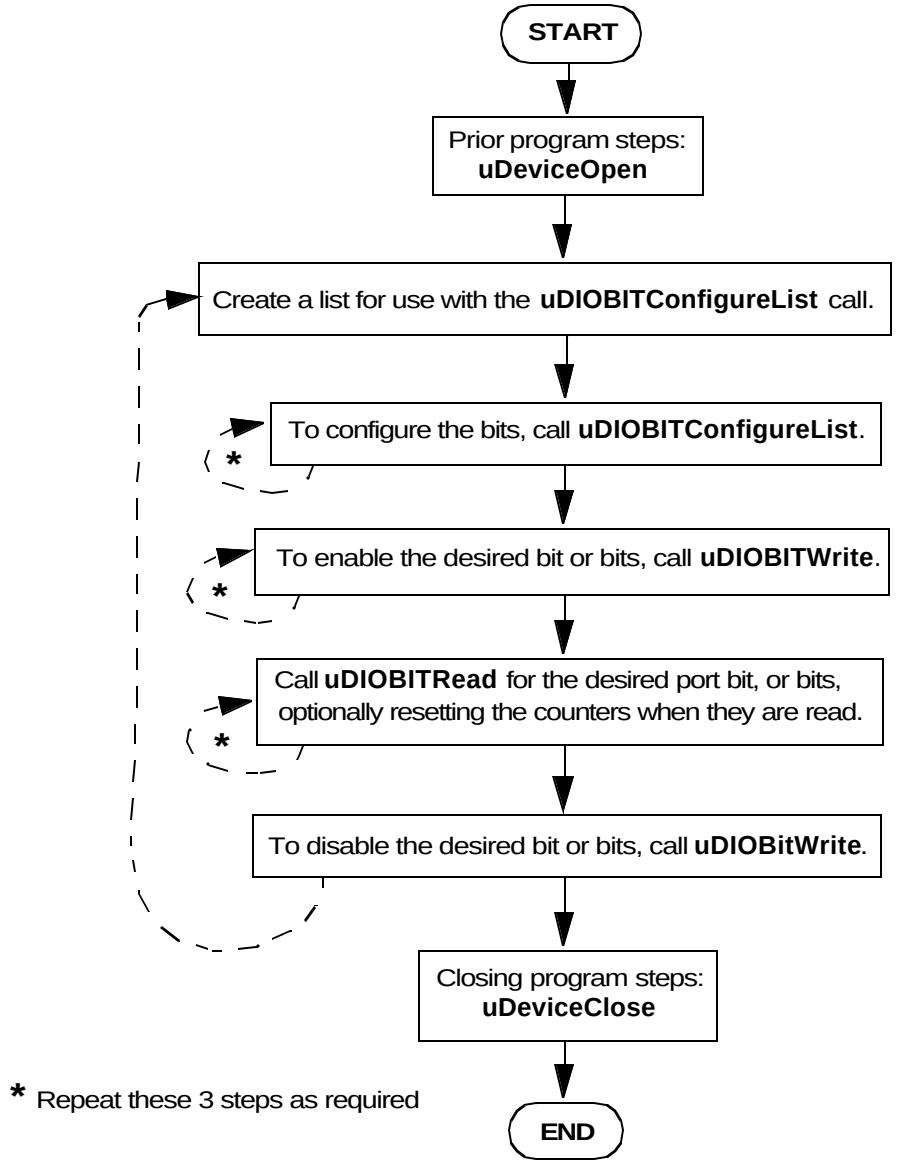
START
Prior program steps:
uDeviceOpen
Create a list for use with the uDIOBITConfigureList call.
To configure the bits, call uDIOBITConfigureList.
*
To enable the desired bit or bits, call uDIOBITWrite.
*
Call uDIOBITRead for the desired port bit, or bits,
optionally resetting the counters when they are read.
*
To disable the desired bit or bits, call uDIOBitWrite.
Closing program steps:
uDeviceClose
END
* Repeat these 3 steps as required

FIGURE 2.3 Digital Trigger Input Mode

2.4.2 Programming Digital Output Bit Modes

This section provides procedures on how to configure each digital output bit mode and write output data.

Output Mode

A sample procedure for programming normal mode digital output bits using U-Link Software Library functions is described below and shown in FIGURE 2.4, *Digital Output Bit Mode.*

Example: Programming Normal Mode Digital Output Bits

1. Create a bit list to be used by the **uDIOBITConfigureList** call.

   Bit list elements use the **uDIOBITListType** structure.

2. For each **uDIOBITListType** element, specify the port that contains the output bit in the *port* field, the actual bit number in the *portbit* field, **DIOBIT_OUTPUT** in the *mode* field, and the desired initial level in the *data* field.

3. Call **uDIOBITConfigureList** to configure the digital I/O bits specified in the bit list. Specify the number of bits in the list in the call's count parameter.

4. To write data to bits configured for normal output mode, call **uDIOBITWrite** for the specified bit, all bits, or selected bits (using a portbit mask).

   Specify the port number in the call's *port* parameter.

The data to be written must be in the buffer pointed to by the *data* parameter. If **ALL_PORT_BITS** or a port bit mask is used, the data buffer must be dimensioned for 8 elements, and data must be stored in the corresponding data buffer elements. The data value indicates the level written to the output bit, logic 1 for high and 0 for low.

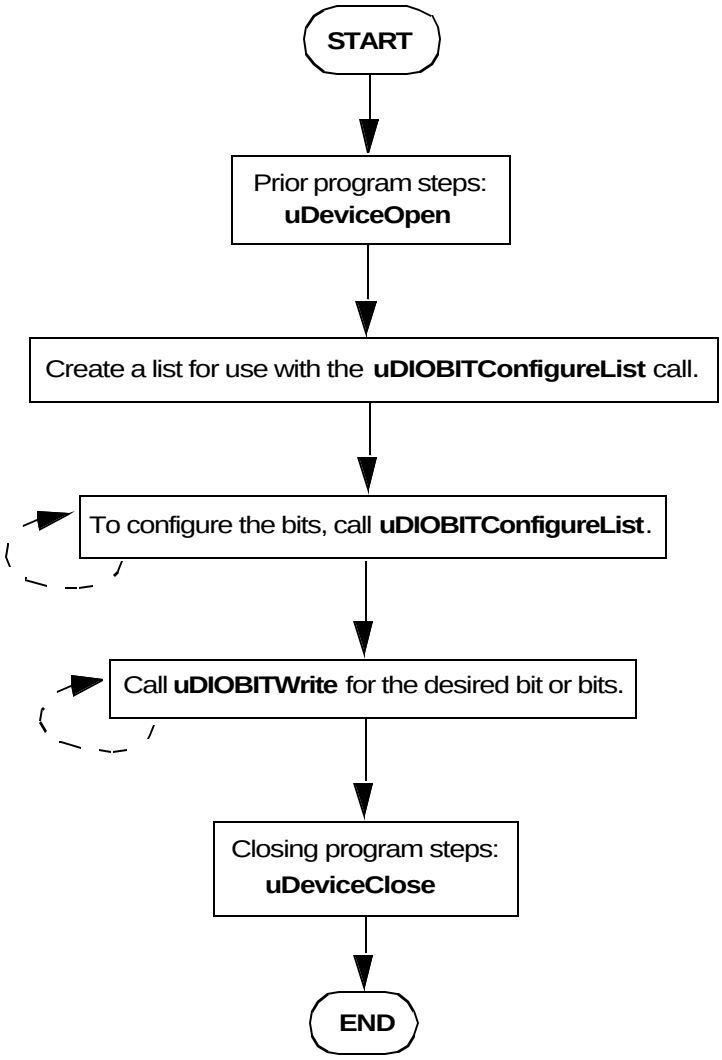

FIGURE 2.4 Digital Output Bit Mode

2.5 Counter Function Calls

Purpose

There is one hardware counter on the UDAS device. The counter is a 16-bit counter with 16 additional bits implemented in software, so that it effectively appears as a 32-bit counter.

Functions

uCTRConfigure configures a counter on a UDAS unit for a specified counter mode.

uCTREnable starts or stops an event counter, or disables a frequency or time measurement.

uCTRRead reads the current count value in event counter mode, or starts a measurement function.

uCTRStatus returns the current status of the event-counting or measurement process.

The **uCTRConfigure** function configures the counter in one of the supported counter modes. It must be called *before* any other counter functions may be used.

> **Note:** If the counter has been previously configured and enabled, it must be disabled before it can be configured again.

Counter Modes

Event Counter

The event counter counts the number of pulses on the input signal from the time the counter is enabled, or is read and reset. Counting occurs when the counter is gated *on* (with the gate input signal) and stops when it is gated *off*. See FIGURE 2.5, *Event Counter Block Diagram*.

If the mode selected is **CTR_MODE_EVENT_HI**, the gate input must be high to enable counting. In mode **CTR_MODE_EVENT_LO**, the gate input is low to enable counting.

The gate input may perform multiple transitions while the counting occurs. The counter may be read at any time and optionally reset. If the counter is not reset, the readings are cumulative; otherwise the result will be counts since the previous reading.

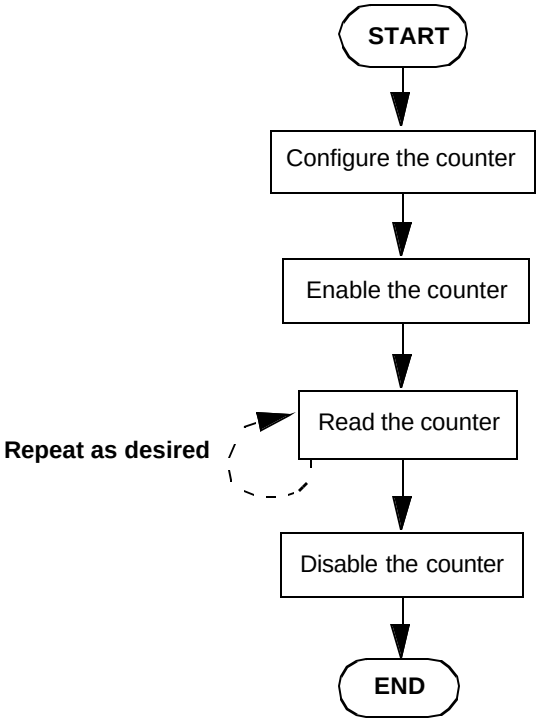

FIGURE 2.5 Event Counter Block Diagram

Period Measurement

In this mode, **CTR_MODE_PERIOD**, the counter measures the duration of one complete period of the input signal. When the counter is read, the data returned is the period measurement in nanoseconds. Because the counter counts 8 MHz clock cycles, the resolution of this measurement is 125 nanoseconds. See FIGURE 2.6, *Period Measurements Block Diagram*.

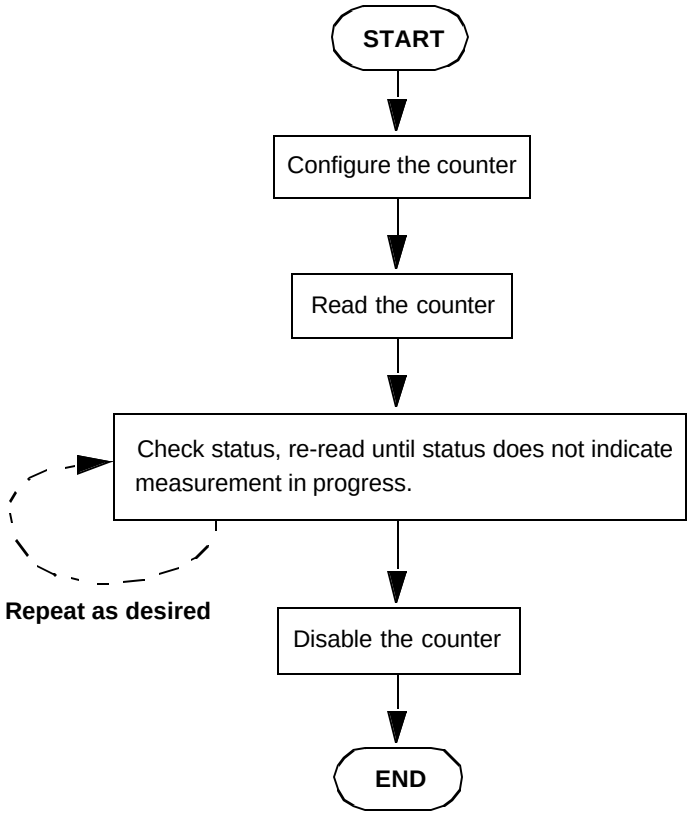

FIGURE 2.6 Period Measurements Block Diagram

Pulse Measurement

The counter measures the duration of the high pulse (**CTR_MODE_PULSE_HI**) or the low pulse (**CTR_MODE_PULSE_LO**) of the input signal in these modes. Since the counter counts 8 Mhz clock cycles, the resolution of this measurement is 125 nanoseconds.

Frequency Measurement

In the **CTR_MODE_FREQ** mode, a gate value (ignored in other modes) must be specified. The counter measures frequency by counting 8 MHz pulses for the duration specified by the gate value, using an internal gate. The frequency measured is returned in Hz (cycles per second).

The gate value determines the precision of the measurement. Valid gate values are: 1 ms, 10 ms, 100 ms, and 1 s. Use **FRQ_GATE_1ms**, **FRQ_GATE_10ms**, **FRQ_GATE_100ms**, and **FRQ_GATE_1s.** See FIGURE 2.7, *Frequency Measurements.*

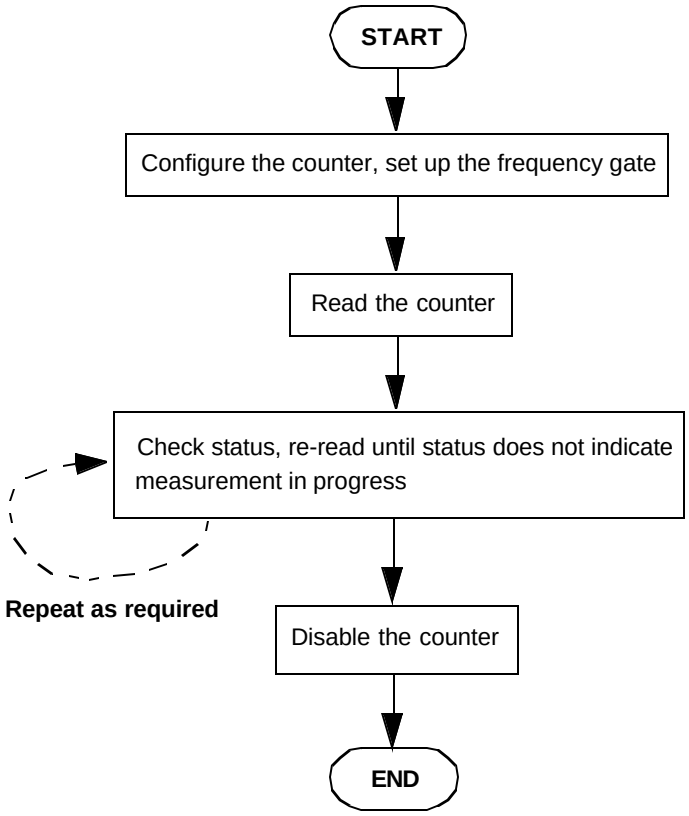
START
Configure the counter, set up the frequency gate
Read the counter
Check status, re-read until status does not indicate measurement in progress
Repeat as required
Disable the counter
END

FIGURE 2.7 Frequency Measurements

2.6 Analog Input Function Calls

Purpose

This section describes how to acquire analog input data with a UDAS-1001E using the U-Link Software Libraries.

U-Link provides two categories of analog input function calls:

- Software controlled analog input.
- Hardware paced high-speed analog input acquisition.

Software controlled analog input allows the gathering of input data from one input channel, or all channels, using software commands. High-speed acquisition allows evenly-spaced sampling of one or more inputs using a time base source and triggering options. High-speed acquisition is described in section 2.7, *High-Speed Analog Input Functions*.

Functions

uAIMAXEnable enables the specified termination panels on AI-MAX 256 channel analog input expansion system.

uAIConfigureList configures, reconfigures, or unconfigures a list of analog input channels for software-controlled input.

uAIRead reads one or more previously configured analog input channels.

The **uAIMAXEnable** function configures the software to use the AI-MAX analog expansion box to access up to 256 channels. Specify the panel numbers in use in the AI-MAX using this function. If used, **uAIMAXEnable** must be called before any analog or digital functions.

To configure a list of analog input channels with a specified gain, input range and input type (single-ended or differential), use the **uAIConfigureList** function. Once configured, the **uAIRead function** call can read the data from these analog input channels. Channels can also be unconfigured and reconfigured using this call.

To read data from a single analog input channel or from all channels, use the **uAIRead** function. Prior to reading data from the analog input, use **uAIConfigureList** to configure the channel or channels. Multiple samples of one or all channels may be read with a single call to **uAIRead**.

Reading multiple samples is similar to successively calling **uAIRead** to read the same channel(s), one sample at a time. (All readings of the first channel are made, then all readings of the second channel, and so forth.) Reading multiple samples this manner does not imply a steady sampling

rate. To sample data at a determined rate, use high-speed analog input function calls. See FIGURE 2.8, *Software Controlled Analog Input Read*.

Example: Two-Dimensional Array

To read data from all channels and multiple samples, the buffer where the data is returned must be properly dimensioned. The following two-dimensional array displays the order that analog input data is returned from the data buffer.

If **ALL_CHANNELS** and **ALL_PANELS** are specified and *count* is equal to *n*, the returned data is organized as shown in Appendix A, *U-Link Software Libraries Function Call Reference*, section A.6.3, *uAIRead*.

This may be implemented by declaring the array for returned data as follows:

- For writing programs in the C language, use the following declaration for an array:
  short data [**MAX_PANELS**] [**CHANNELS_PER_PANEL**] [**count**];
- If AI-MAX is not used, the panel number is always 0. Omit the **MAX_PANELS** dimension.
- If some panels are not used, or if some of the channels are not configured, these elements still require allocated space in the returned data array.
- The parameters specified must not return more than 1021 values. To read more values, make additional calls to **uAIRead**.

The unipolar data is adjusted to an unsigned number format before the data is returned to the application so the data array for unipolar data should be declared as unsigned short.

Analog input data is returned in A/D (Analog/Digital) counts.

- If the bipolar input range (±10 V or ±5 V) is used, the data values range from -32768 to 32767.
- If the unipolar input range (0-10 V) is used, the data values range from 0 to 65535.
- Use the **uCountsToVolts** function to convert counts data to floating point voltage values.

Programming Procedures

The **uAIConfigureList**, **uAIRead**, and **uAIMAXEnable** call descriptions in Appendix A, *U-Link Software Libraries Function Call Reference*, provide all of the required information for using these functions. This includes syntax, a functional description, and a list of parameters and their possible values. Refer to Appendix A for gain, range, input type, and other codes related to these functions. It may be helpful to refer to the analog input sample programs provided with U-Link for real code examples.

Using the **uAIConfigureList** and **uAIRead** calls, a read operation can be performed on any *individual* analog input channel or on *all* analog input channels.

- When a configuration includes an AI-MAX device for system expansion, use the **uAIMAXEnable** call.

A flow chart for this procedure is shown inFigure 2.8, *Software Controlled Analog Input Read*.

1. Create a channel list according to the format of the **uAIListType** analog input channel list structure.
2. Set the analog input configuration for each channel to the desired gain, range, input type (single-ended or differential), and optional zero-reference.
3. Call **uAIConfigureList**, reference the channel list created in step 1, and specify the count (the number of channels on the list).
4. Allocate a buffer for storing the analog input data that will be acquired.

   Allocate the local buffer used for data storage as static or local data, or using malloc or another appropriate run-time library or Windows API function. Unless an AI-MAX device is used, the maximum required buffer size in bytes is: **Buffer size (in bytes) = (2 bytes/sample) x 16 channels x samples**

   > **Note:** The maximum number of single-ended channels is 16 (unless AI-MAX is used) and each reading (or sample) uses 2 bytes. *Samples* are the number of samples per channel you specify to uAIRead in the next step.

5. To read a specific analog input channel or all of the analog input channels on the UDAS unit, call **uAIRead**.
6. Specify the number of samples per channel to be returned in the call's samples parameter. The data is returned in *counts*.
7. To convert the returned count data to floating point voltage values for further processing in the program, use the **uCountsToVolts** utility function.

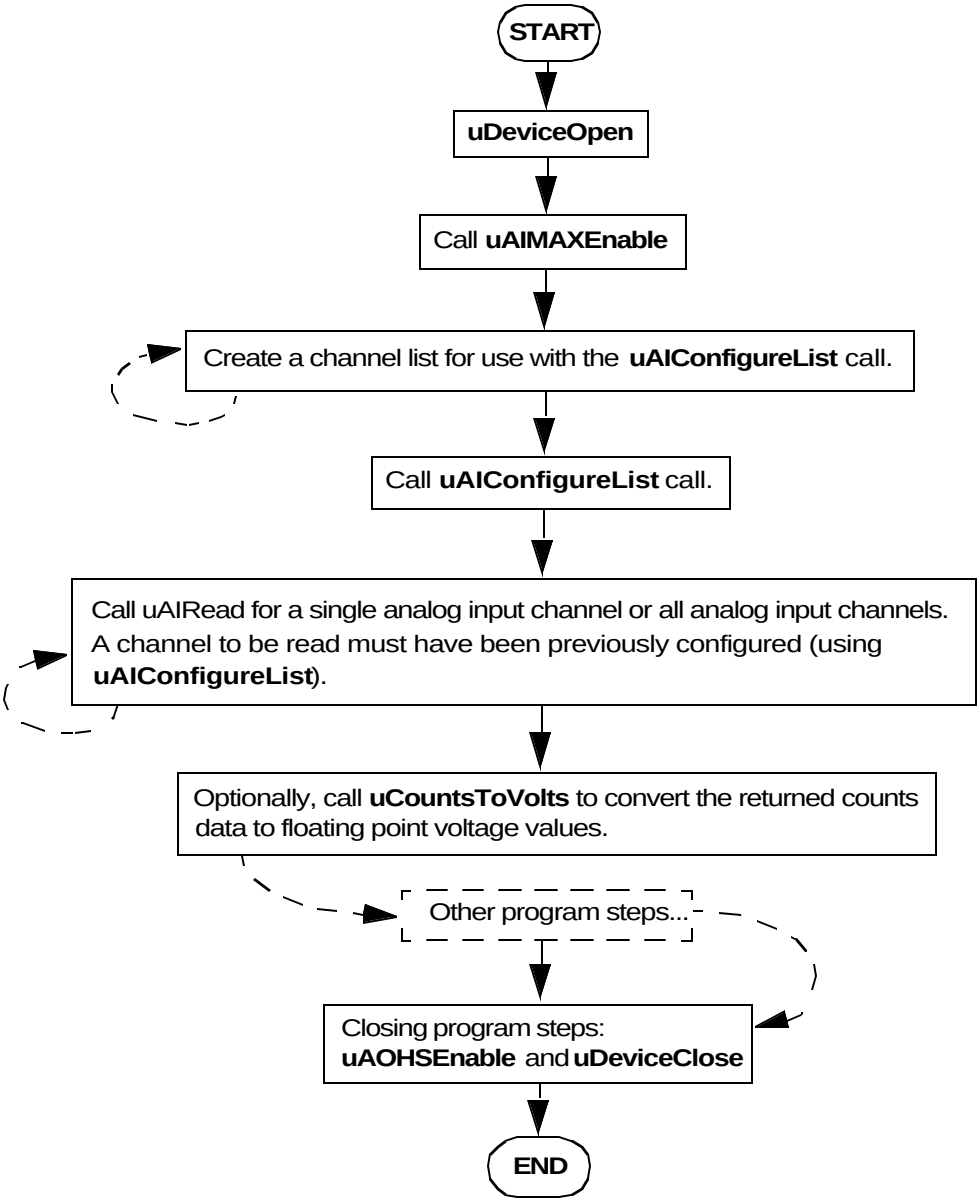
START
uDeviceOpen
Call uAIMAXEnable
Create a channel list for use with the uAIConfigureList call.
Call uAIConfigureList call.
Call uAIRead for a single analog input channel or all analog input channels. A channel to be read must have been previously configured (using uAIConfigureList).
Optionally, call uCountsToVolts to convert the returned counts data to floating point voltage values.
Other program steps...
Closing program steps: uAOHSEnable and uDeviceClose
END

FIGURE 2.8 Software Controlled Analog Input Read

2.7 High-Speed Analog Input Functions

A high-speed analog input process uses a pacer, either the internal rate generator or an externally supplied signal, to time the acquisition of analog samples. One or more channels may be sampled. The process may be started by a trigger event or under program control. It may be stopped after acquiring a set amount of data, after a trigger event, or under program control.

Purpose

This section includes functions used for high-speed analog input processes.

Functions

uAIHSConfigureList configures a high-speed analog input process.
uAIHSEnable enables or disables a high-speed analog input process.
uAIHSReturn returns data from device buffers to a host buffer managed by the driver.
uAIHSRead reads the data acquired by a high-speed analog input process and returns it to the user's application.
uAIHSStatus obtains the status and buffer information of a high-speed analog input process.

To configure a list of analog input channels for a high-speed analog process, use the function **uAIHSConfigureList**. This call also specifies the start and stop methods of the acquisition and the pacer source and optional trigger source. For more information on Rate Generators, see *Rate Generator Function Calls,* on page2-36. For additional information on Trigger functions, see *Trigger Function Calls*, on page 2-39.

- The channel list is specified to the **uAIHSConfigureList** call and uses the same format as the channel list specified to the **uAIConfigureList** call. The **uAIHSConfigureList** call's list parameter uses the **uAIListType** channel list structure and the call's count parameter tells the function the number of channels you want to configure.
- After configuring the analog input channels and configuring the pacer and trigger sources (through appropriate U-Link function calls), data from these analog input channels can be read by enabling the high-speed process using **uAIHSEnable**, then calling the **uAIHSReturn** and **uAIHSRead** function calls.
- To obtain status information about the acquisition process, use the **uAIHSStatus** function.

General Rules for Setting up a Channel List

When setting up a channel list, there are some general rules to follow that are required by the operation of UDAS-1001E hardware in high-speed acquisitions.

- The channels in the list must be specified in consecutive order, from channel 0 to channel N, with no gaps, where N is the *last* channel in the list.
- Each channel in the list must have the same configuration settings for gain, input voltage range and input type (single-ended or differential).
- A zero-reference *can not be used in high-speed acquisitions*. Set the zero-reference field of **uAIListType** to **AI_NO_ZERO**.

The **uAIHSConfigureList** call returns a value in its clustersize parameter indicating the number of bytes required to store the input data for one pass through the channel list. This information is used by the **uAIHSEnable** call to allocate a data buffer in the host memory for returned data.

> **Note:** The amount of data acquired may be limited by your system resources.

2.7.1 Pacers and Triggers

High-speed analog input data acquisition requires the use of a pacing signal or timebase. Pacing signal pulses initiate A/D conversions at regularly spaced intervals. After each A/D conversion on a UDAS-1001E, the data is transferred to a local buffer and the converter input is switched to the next channel in the list.

Acquisition continues until some terminating condition occurs. The data acquired may be transferred from the device to the host in response to **uAIHSRead** or **uAIHSReturn**.

Pacers

The UDAS-1001E supports two pacer sources:

- On-board Analog Input Rate Generator
- External Input (at pin 47 of its 50-pin I/O connector)

The desired pacer source is specified to the **uAIHSConfigureList** call through its *pacer* parameter. A crystal-based oscillator on the UDAS unit provides the Analog Input Rate Generator with a stable clock source.

For information on programming the rate generator's output frequency, see *Rate Generator Output Frequency*, on page 2-36 and *Rate Generator Output*, on page 2-41.

> **Note:** If the External Input is selected as the pacer, a TTL level signal on this input is required to start A/D conversions using a low-to-high or high-to-low transition. The signal should be supplied via the external input line (EXT IN) on the UDAS unit's I/O connector.

Triggers

A trigger is useful for capturing data based on some external event. Using a UDAS unit, a trigger can be used to start an acquisition based on an analog input voltage level-crossing or a digital signal connected to the External Input line on the unit's I/O connector. The desired trigger source (or none) is specified to the **uAIHSConfigureList** call through its trigger parameter.

The analog trigger on the UDAS unit uses analog output channel 0 to set a threshold for comparison with any analog input channel being measured. Rising and falling edge detection modes are programmable.

See *Trigger Function Calls*, on page 2-39, for information on programming the analog trigger function. If the External Input is selected as the trigger, a TTL low-to-high or high-to-low transition can be specified as the trigger event.

> **Note:** The same external source can be used for pacer and trigger, if desired.

Pacer and trigger sources are automatically connected to the desired target circuit inputs (A/D start convert, A/D trigger) and the trigger is enabled when **uAIHSEnable** is called.

> **Note:** If the rate generator is used as a pacer, you must explicitly enable the rate generator.

Start and Stop Modes

High-speed start and stop methods are specified in the **uAIConfigureList** call's start and stop parameters.

Start modes supported by the UDAS units include:

- Start on Command
- Start on Trigger

Supported stop modes include:

- Stop on Terminal Count
- Stop on Trigger with Delay
- Stop on Command

The following paragraphs discuss the operation of each start and stop mode.

Start on Command

When specifying the *Start on Command* mode in **uAIHSConfigureList**, data acquisitions begin when the **uAIHSEnable** function, with its enable parameter set to a value of 1, is called.

Start on Trigger

When specifying the *Start on Trigger* mode, data acquisition begins when the programmed trigger condition occurs (after the process is enabled by a call to **uAIHSEnable** with its enable parameter set to a value of 1).

Stop on Terminal Count

When specifying the *Stop on Terminal Count* mode, acquisitions stop when the programmed number of samples per channel are captured.

- Use the **uAIHSStatus** call to detect the end of a terminal count process.

Stop on Command

When using the *Stop on Command* mode, data acquisitions stop when the **uAIHSEnable** function is called with its enable parameter set to a value of 0. For programs that contain a *Stop on Command* mode, data is acquired in a *circular buffer* on the host. The number of channels in the channel list (count parameter in the **uAIHSConfigureList**) determines the size of this data buffer and the maximum number of samples to read per channel (count parameter in the **uAIHSEnable**).

- If the total amount of data acquired will exceed 64K bytes, use the **uAIHSReturn** call to return data to the host during the process. Data is written to a circular buffer until the process stops.

Stop on Trigger with Delay

Acquisitions stop when the delay count is reached following the trigger event. Data is acquired to a circular buffer until the process stops. If the total amount of data acquired is to exceed 64K bytes, use the **uAIHSReturn** call to return data to the host during the process. Data is written to a circular buffer until the process stops.

> **Important:** Unless you plan to configure a process once and restart it multiple times, free the local data buffer by calling **uAIHSEnable** with its enable parameter set to 0 and its free parameter set to logic 1 when the process terminates. If using the Stop on Command option, do not free the buffer until the data has been read from the buffer.

For example, for a digital oscilloscope application using a process configured for *Start on Trigger* and *Stop on Terminal Count* modes, call **uAIHSConfigureList** once, followed by repeated calls to **uAIHSEnable** and **uAIHSRead**. When the process is no longer needed, call **uAIHSEnable** to terminate the process and free the UDAS unit's data buffer.

Reading High-Speed Data

As data is acquired, buffers are queued on the UDAS device for the **uAIHSReturn** function. The host reads data from the UDAS and stores it in allocated memory. The **uAIHSRead** command returns data from the host memory to the application.

The analog input data is returned in *counts*.

- For bipolar input ranges (±5 V or ±10 V), the *counts* format is -32768 to 32767.
- For the unipolar input range (0 - 10 V), the *counts* format is 0 to 65535.
- To convert counts data to floating point voltage values, use the **uCountsToVolts** utility function that is supplied with U-Link. See Appendix A, *U-Link Software Libraries Function Call Reference* for a listing of the utilities.

The **uAIHSReturn** function returns data from the UDAS during a high-speed acquisition. If more than 64K bytes will be acquired, you must call **uAIHSReturn** to copy the data into a *circular* buffer on the host while acquisition is proceeding. If you use the *Stop on Trigger with Delay* option or the *Stop on Command* option, a large amount of data may be acquired prior to the stop event. Although this data may ultimately be discarded, it must be returned to the host or the device's buffer space will overflow, resulting in an error.

Programming Procedures

The function call descriptions in Appendix A, *U-Link Software Libraries Function Call Reference*, provide all of the necessary information for using high-speed analog input functions. This includes syntax, a functional description, and a list of parameters and their possible values. Refer to Appendix A for valid parameter values and any codes related to these functions. The high-speed analog input sample programs provided with U-Link of real code examples are a helpful resource. See Figure 2.9, *High-Speed Analog Input Acquisition*.

The host buffer is automatically allocated for storing the analog input data when the data acquisition begins.

- To allocate a local buffer for data storage, use a malloc or other appropriate run-time library or Windows API function, or declare a local or static buffer.
- The size of a buffer to hold one cluster of returned data (one complete cycle through all channels read) is given by *clustersize*. Multiply this size by the number of clusters to be returned to your application at once.

Reading Analog Input Data

The steps you choose to read the analog input data and end the high-speed process are dependent on the selected start and stop modes:

Start on Command and Stop on Command

For this mode combination, the analog input acquisition starts when the **uAIHSEnable** function is called. Analog input data is continually acquired and queued on the UDAS unit until the process is terminated by a call to **uAIHSEnable** with the enable parameter set to 0.

- To return data to the host circular buffer until the **uAIHSEnable** is called to stop the process, call **uAIHSReturn**.

  Calling **uAIHSReturn** returns data from the device to the host buffers. If this function is not called, data is queued. Calling **uAIHSRead** returns data from the device *only* if there is no data to be read from the host buffers; the function returns the number of clusters actually read.

  After the process is stopped, make a final call to **uAIHSReturn** to return the last data acquired.

Start on Command, Stop on Terminal Count

For this mode combination, the analog input acquisition starts when the **uAIHSEnable** function is called. Analog input data is acquired and queued until the number of requested samples per channel (the terminal count) have been acquired. If the amount of data acquired will exceed 64K bytes, you must call **uAIHSReturn** to return data from the device while the acquisition is occurring.

Start on Trigger, Stop on Terminal Count

For this mode combination, the process is enabled when the **uAIHSEnable** function is called, and data acquisition starts when the trigger condition occurs. After the trigger event occurs, analog input data is acquired and queued on the UDAS until the number of requested samples per channel (the terminal count) have been acquired. If the amount of data acquired will exceed 64K bytes, you must call **uAIHSReturn** to return data from the device while the acquisition is occurring.

To verify that the trigger has occurred, test the status word from a call to **uAIHSStatus** or **uAIHSReturn**.

For information on reading the data and terminating the high-speed process refer to the *Start on Command, Stop on Terminal Count* information.

Start on Command, Stop on Trigger with Delay

For this configuration, the number of clusters acquired is undetermined, and the buffer may be overwritten several times while waiting for the trigger. The maximum number of clusters available is defined by the value of *count*. If the value for *delay* is less than the value for *count*, the final buffer contains both pre-trigger and post-trigger data. If the trigger occurs immediately after the start command, the buffer may not be full. If the *delay* and *count* values are equal, the buffer contains only post-trigger data starting immediately after the trigger event occurs.

Programming Procedures

1. Initialize U-Link with the **uDeviceOpen** call.
2. Create a channel for use with the **uAIHSConfigureList** call.
3. Configure the pacer source. When using the External Input, ensure the external pacer source is connected. When using the analog input rate generator, call **uRGConfigure** to configure the generator mode and frequency.
4. Call the **uAIHSConfigureList** function to configure the high-speed process and to obtain the returned clustersize value for later use.
5. Allocate a buffer to be used for data returned by **uAIHSRead**.
6. If this is a Start on Trigger process, configure the trigger source. When using the analog trigger circuit, call **uTRIGConfigure** to configure the threshold and mode. If the external trigger source is used, verify that it is connected.
7. Start the external pacer source or enable one rate generator by calling **uRGEnable**.
8. Call **uAIHSEnable**, with enable = 1 to enable the high-speed process.

   When this call is made (with enable = 1), the pacer and trigger sources are connected to internal A/D circuitry.
9. Depending on the start mode, data acquisition immediately starts after **uAIHSEnable** is called (*Start on Command* mode), or after a trigger event (*Start on Trigger* mode) occurs.
10. Call **uAIHSReturn** as required by the application.
11. Call **uAIHSEnable** if needed to stop the acquisition.
12. Call **uAIHSRead** to return data to the application.

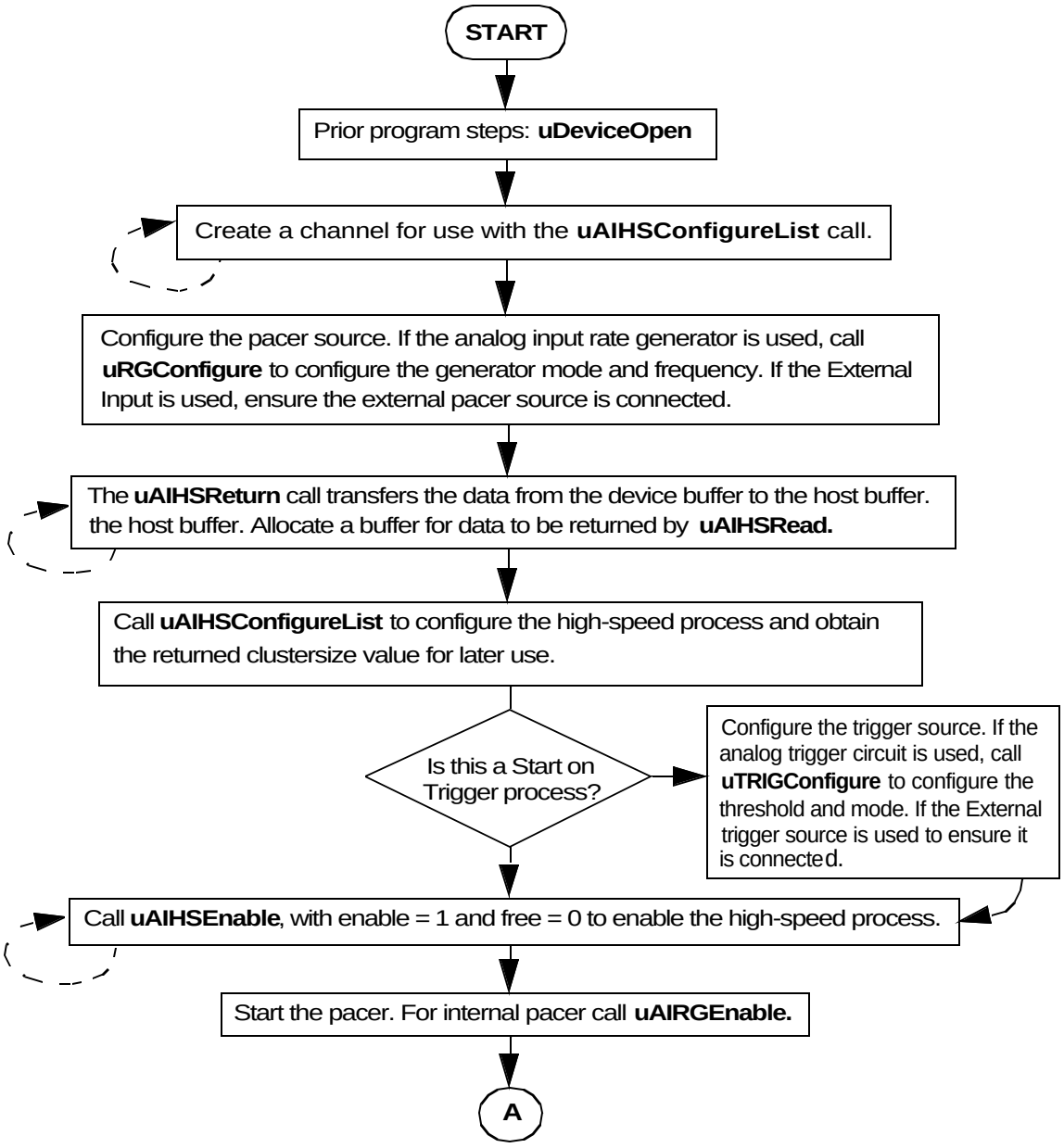
START
Prior program steps: uDeviceOpen
Create a channel for use with the uAIHSConfigureList call.
Configure the pacer source. If the analog input rate generator is used, call uRGConfigure to configure the generator mode and frequency. If the External Input is used, ensure the external pacer source is connected.
The uAIHSReturn call transfers the data from the device buffer to the host buffer. the host buffer. Allocate a buffer for data to be returned by uAIHSRead.
Call uAIHSConfigureList to configure the high-speed process and obtain the returned clustersize value for later use.
Is this a Start on Trigger process?
Configure the trigger source. If the analog trigger circuit is used, call uTRIGConfigure to configure the threshold and mode. If the External trigger source is used to ensure it is connected.
Call uAIHSEnable, with enable = 1 and free = 0 to enable the high-speed process.
Start the pacer. For internal pacer call uAIRGEnable.
A

FIGURE 2.9 High-Speed Analog Input Acquisition

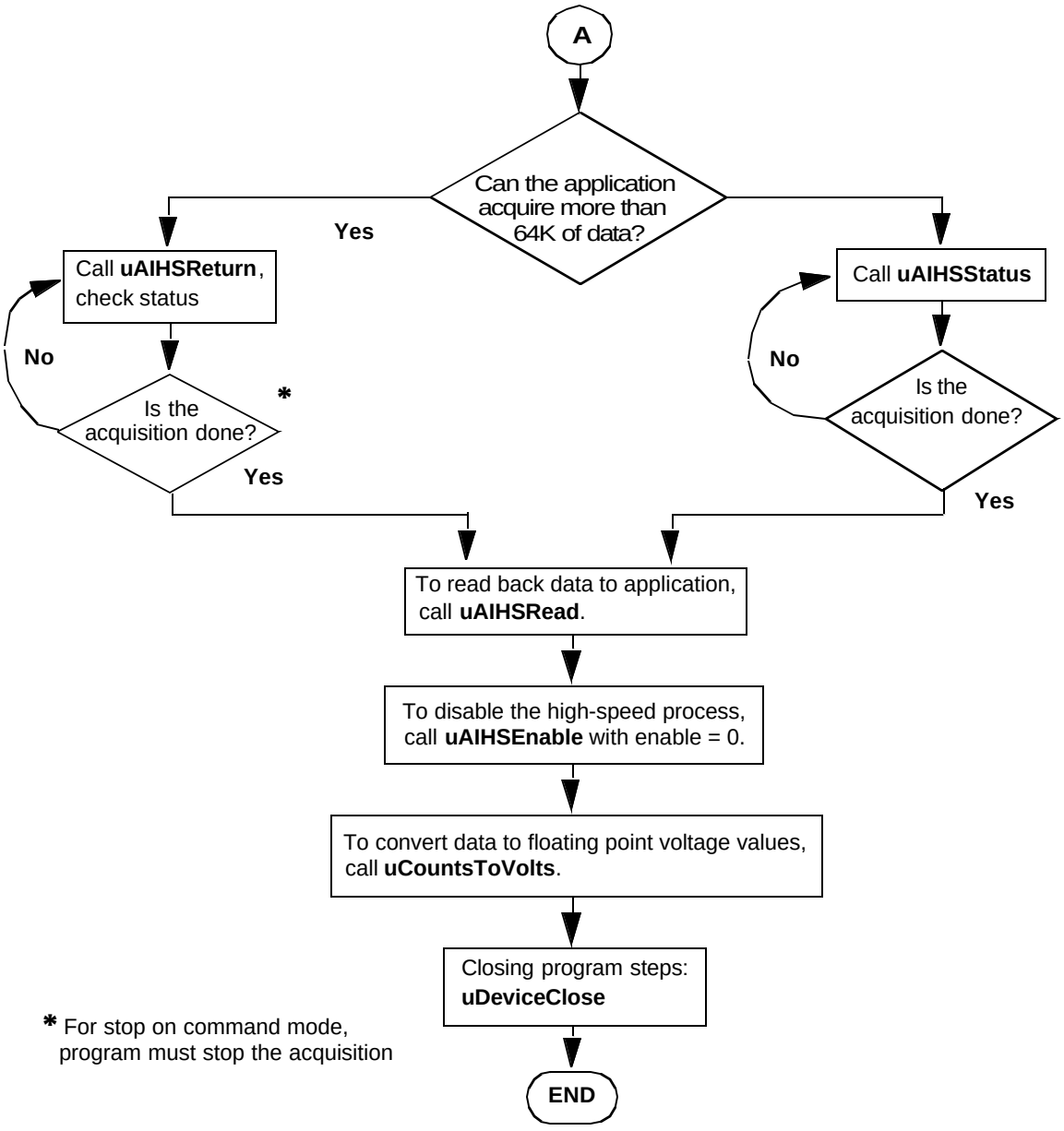
A
Can the application acquire more than 64K of data?
Yes
Call **uAIHSReturn**, check status
No
Is the acquisition done?
*
Yes
Call **uAIHSStatus**
No
Is the acquisition done?
Yes
To read back data to application, call **uAIHSRead**.
To disable the high-speed process, call **uAIHSEnable** with enable = 0.
To convert data to floating point voltage values, call **uCountsToVolts**.
Closing program steps: **uDeviceClose**
END
* For stop on command mode, program must stop the acquisition

FIGURE 2.9 High-Speed Analog Input Acquisition (continued)

2.8 Analog Output Function Calls

Purpose

This section describes how to output analog data with UDAS-1001E units using the U-Link Software Libraries. A UDAS-1001E unit provides two voltage-output channels with 12-bit resolution. U-Link functions support the configuration of analog output channels and transfer of data under software control.

Functions

uAOConfigureList configures a list of analog output channels.

uAOWrite writes data to previously configured analog output channels.

To configure a list of analog output channels, use the **uAOConfigureList** function. Once configured, use the **uAOWrite** function to write data to these analog output channels.

The channel list is specified through two **uAOConfigureList** parameters: *list* and *count*.

The *list* parameter uses a defined analog output channel list structure called **uAOListType**. The *count* parameter tells the function the number of channels that you want to configure (1 or 2).

To write data to a previously configured analog output channel (0 or 1), or to all configured analog output channels (**ALL_CHANNELS**), use the **uAOWrite** call.

- The data buffer holds the data to be written to the specified analog output(s). Specify data values in counts.
- To convert floating point voltage values to D/A counts for the data buffer, use the **uVoltsToCounts** utility function.

Since the analog converter accepts 12 bits, the actual top range is 32752 (full scale minus one LSB). If the low four bits of the data are non-zero, they are truncated.

2.9 Analog Output High-Speed Functions

Purpose

This section explains how to output high-speed analog data with UDAS units using the U-Link Software Libraries.

Summary of Analog Output High-Speed Functions

High-speed analog output is supported for two voltage output channels with 12-bit resolution on a UDAS unit. Analog output is paced by the on-board analog output rate generator or optionally by the pacer for a simultaneous analog input process.

The analog output FIFO is capable of operating as a circular buffer to generate a limited waveform for one or both channels under hardware control. Longer waveforms can be output under software control (at a lower maximum rate).

Functions:

uAOHSConfigureList configures a list of analog output channels.
uAOHSEnable enables the analog output process and transmits the data to be output, or dismisses the analog output process.
uAOHSWrite downloads a segment of continued analog output waveform to the device.
uAOHSStatus returns the output status. The output terminates when there is no more data is available on the device.

To configure a list of analog output channels, use the **uAOHSConfigureList** function. Once the channels are configured, waveforms are output by the **uAOHSEnable** and **uAOHSWrite** function calls.

The channel list is specified through two **uAOHSConfigureList** parameters: *list* and *count.*

The *list* parameter uses a defined analog output channel list structure called **uAOListType**. The *count* parameter tells the function the number of channels that you want to configure (1 or 2).

The list that you create is an array of up to two elements of type **uAOListType**. The **uAOHSConfigureList** call returns the clustersize, the size of data in bytes for a single pacer tick.

The **uAOHSEnable** command enables or disables the analog output process and transmits the data to be output. If the data pointer passed is *NULL*, and *enable* is true, the command resumes output of a waveform already transferred and present in the FIFO. If there is no waveform present, an error occurs.

To convert the desired voltage to count data for analog output, use the **uVoltsToCounts** call.

If the *repeat* flag is specified *TRUE*, the data provided by the **uAOHSEnable** function is output repeatedly. If the amount of data for one cycle is small enough to fit into the FIFO, the waveform will be automatically repeated under hardware control. Larger amounts of data are stored in UDAS local memory and continuously reloaded into the FIFO.

If the *repeat* flag is **FALSE**, the data provided is output once. The output proceeds continuously as long as the **uAOHSWrite** function provides more data before the FIFO empties. When the FIFO empties, the output terminates and the **uAOHSStatus** or **uAOHSWrite** function may detect an error condition.

The **uAOHSWrite** call downloads a continued output waveform to the device. The output process must be previously started with an initial data buffer, using the **uAOHSEnable** function.

Use the **uAOHSStatus** function to detect premature termination of a waveform output under software control, or to verify the completion of a non-repeated waveform.

Data for high-speed output to two channels is stored with the channels alternating. Declare the array for data to be passed by **uAOHSEnable** or **uAOHSWrite** as:

short data [count] [channels];

Programming Procedures

The call descriptions in Appendix A, *U-Link Software Libraries Function Call Reference* provide all of the information for using these functions, including syntax, functional descriptions, parameter lists and their possible values.

The analog output process occurs with the following steps.

A flow chart for this procedure is given in FIGURE 2.10, *Writing High-Speed Analog Output Data*.

1. Disable any previously configured analog output by using **uAOHSEnable**.
2. Set the analog acquisition parameters via **uAOHSConfigureList**.
3. If the pacer is the Rate Generator, set its parameters via **uRGConfigure**.
4. To synchronize with a data acquisition process, configure the data acquisition and its pacer and trigger.
5. Start the data output process by calling **uAOHSEnable**. This call enables the Rate Generator and transmits the initial data to the output process.
6. Starts the data acquisition process if it was configured in Step 4.

7. If a longer waveform is needed, call **uAOHSWrite** to transmit additional data to the output process.
8. Check the status of the application using the **uAOHSStatus** function.
9. To stop the output, call **uAOHSEnable**. If repeat mode is not used, the output process will stop automatically.

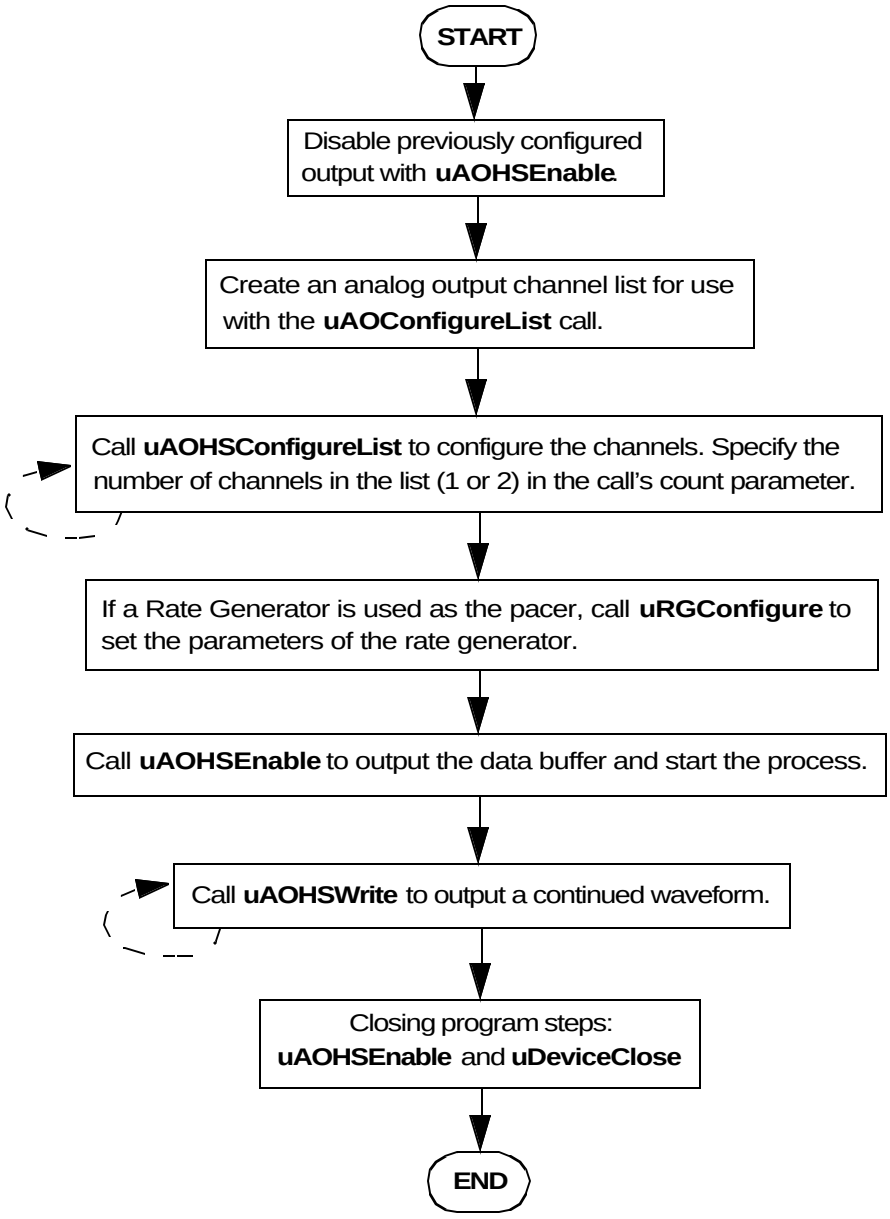
START
Disable previously configured output with **uAOHSEnable**.
Create an analog output channel list for use with the **uAOConfigureList** call.
Call **uAOHSConfigureList** to configure the channels. Specify the number of channels in the list (1 or 2) in the call's count parameter.
If a Rate Generator is used as the pacer, call **uRGConfigure** to set the parameters of the rate generator.
Call **uAOHSEnable** to output the data buffer and start the process.
Call **uAOHSWrite** to output a continued waveform.
Closing program steps:
uAOHSEnable and **uDeviceClose**
END

FIGURE 2.10 Writing High-Speed Analog Output Data

2.10 Rate Generator Function Calls

Purpose

There are two 24-bit hardware counters on the UDAS 1001E which implement rate generators to provide a source of evenly-spaced pulses for pacing A/D and D/A conversions in high-speed processes. The output of a rate generator can also be used as a general purpose source of clock pulses. A rate generator's output is available at the UDAS unit's I/O connector, selectable between AI and AO. Use the U-Link Software Libraries to program the rate generators for a specific output frequency.

Functions

uRGConfigure configures a rate generator channel with a frequency expressed in terms of a 24-bit count of 8 MHz pulses count values.

uRGEnable enables or disables a rate generator channel.

Rate Generator Output Frequency

The rate generator's output frequency is the result of dividing the on-board 8 MHz timebase frequency by a 24-bit integer *count* value.

The output frequency is given by the formula:

F = 8 MHz / count

where *count* can range from 2 to 16777216.

From the formula shown above, the output frequency can range from a maximum of 4 MHz to a minimum of approximately 0.4 Hz.

> **Note:** The analog input circuit and the waveform output circuit support a maximum sampling rate of 100 kHz.

To obtain the count value for the desired output frequency, use the **uRGFrequencyToCounts** utility function.

Prior to configuration or reconfiguration, it is necessary to disable the rate generator by using the **uRGEnable** call.

To enable the rate generator, set the **uRGEnable** call's enable flag to 1. The analog output rate generator is automatically enabled when you start the high-speed analog output process. The analog input rate generator must be enabled prior to enabling the high-speed analog input process.

Use the **uSETOption** function to select the rate generator to be connected to the termination output.

Programming Procedures

A sample procedure for programming a rate generator using U-Link Software Library functions is described below and shown in FIGURE 2.11, *Rate Generator Programming*. For detailed information on each of the rate generator calls, including a description and a parameter list, see Appendix A, *U-Link Software Libraries Function Call Reference.*

Example: Programming a Rate Generator

1. Initialize U-Link with the **uDeviceOpen** function.
2. To calculate the count value for the desired output frequency, call the **uRGFrequencyToCounts** utility function.
3. To set the output frequency for the rate generator channel, call **uRGConfigure**. The count value, obtained in Step 2. programs the desired generator frequency.
4. If the rate generator is not being used for a high-speed analog output process, call **uRGEnable** with its enable flag set to 1 to enable the rate generator for its programmed mode. If the rate generator is being used in a high-speed analog output process, the rate generator is enabled when the **uAOHSEnable** call is used to start the process.
5. When the rate generator is no longer needed, disable it using the **uRGEnable** call.
6. Close the program using the **uDeviceClose** call.

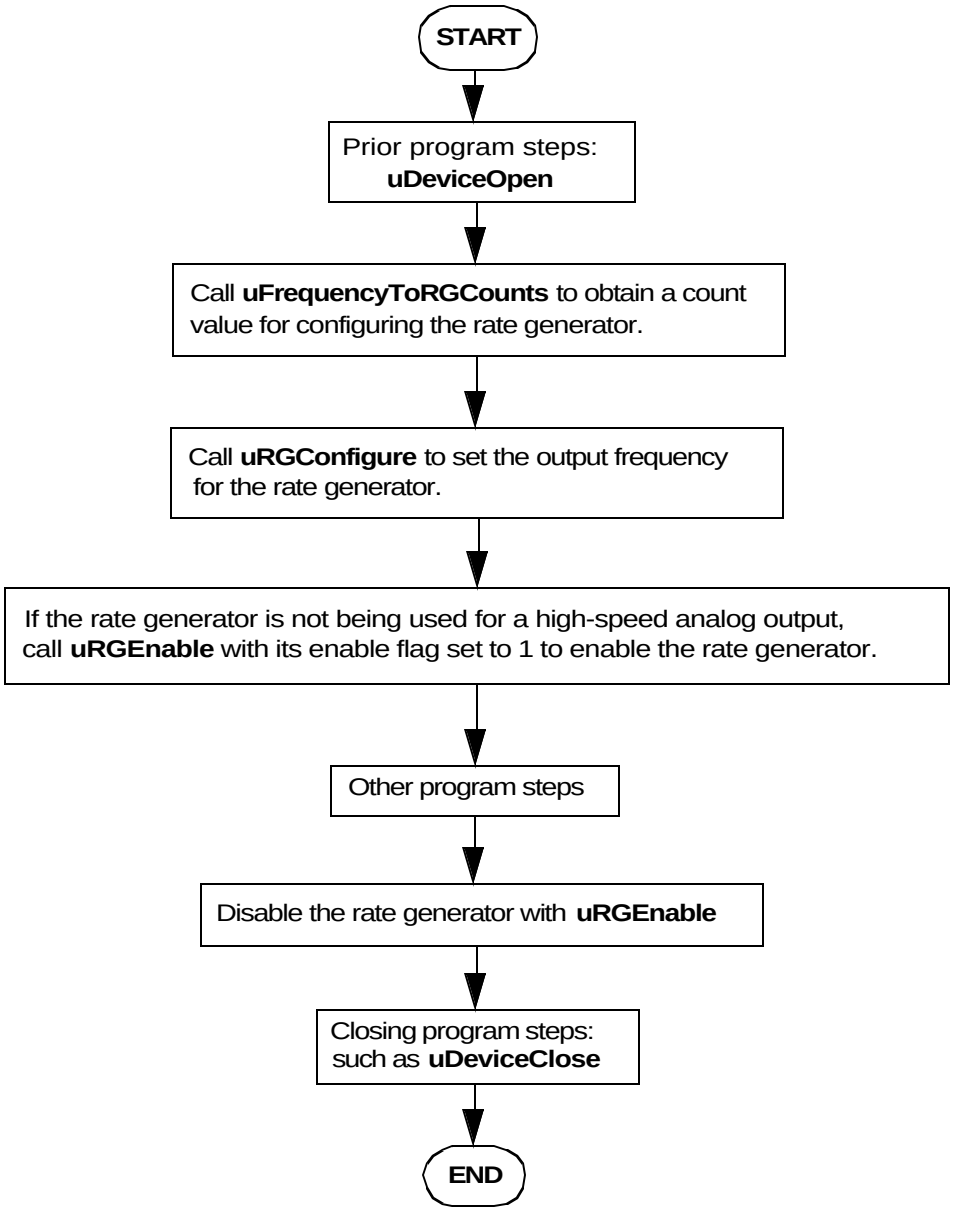
START
Prior program steps:
uDeviceOpen
Call uFrequencyToRGCounts to obtain a count value for configuring the rate generator.
Call uRGConfigure to set the output frequency for the rate generator.
If the rate generator is not being used for a high-speed analog output, call uRGEnable with its enable flag set to 1 to enable the rate generator.
Other program steps
Disable the rate generator with uRGEnable
Closing program steps:
such as uDeviceClose
END

FIGURE 2.11 Rate Generator Programming

2.11 Trigger Function Calls

Purpose

This section discusses how to configure an analog input channel as a trigger for a high-speed analog input process on a UDAS device. The analog trigger circuit on a UDAS unit uses any analog input channel as an input to an internal comparator circuit. When the input voltage on a selected input channel crosses the threshold voltage, it generates a trigger signal. The threshold level is set using analog output channel 1. When using an analog trigger, do not use analog output channel 1 for any other purpose.

Using a trigger, data acquisition can be controlled based on some external event. When the analog trigger circuit is the trigger source, a rising or falling input voltage crossing a programmable threshold level activates the trigger signal.

Function

uTRIGConfigure: Configures an analog trigger channel for operation.

The function call **uTRIGConfigure** configures the analog trigger on a UDAS unit for a mode and a threshold. In the call's mode parameter, specify the trigger mode, trigger on rising edge (**TRIG_LOtoHI**) or trigger on falling edge (**TRIG_HItoLO**). The threshold value, is given in counts -32768 to 32767 corresponding to the ±10 V voltage range of analog output channel 1.

> **Note:** Use the uVoltsToCounts utility function to convert floating point voltage values to count values for the threshold.

- When the high-speed analog input process using the trigger is enabled by an **uAIHSEnable** call, the trigger output is automatically connected to the A/D trigger input.
- The trigger is automatically reset by calling **uAIHSEnable** (with enable = 1). A process can be restarted as many times as desired. Use this technique to create a digital oscilloscope application with an analog trigger.
- Any channel may be used as the trigger channel so long as it is included in the programmed channel list. If the high-speed acquisition process uses the *start on trigger* mode, the sampled channels appear in a rotated order if the trigger channel is not channel 0.

- Since the analog output circuit has a range of ±10 V and a resolution of 12 bits, an approximate ±5 mV change at the trigger input can be detected.

> **Note:** The trigger channel is compared with the threshold after applying the programmed gain.

Programming Procedures

This section includes a sample procedure for programming the analog trigger circuit using U-Link Software Library functions and is shown inFigure 2.9, *High-Speed Analog Input Acquisition*. See Appendix A, *U-Link Software Libraries Function Call Reference* for detailed information on each of the function calls, including a description and a parameter list.

Example: Programming the Analog Trigger Circuit with U-Link Functions

1. To obtain the count value for the desired trigger threshold voltage, call the **uVoltsToCounts** utility function.
2. To set the mode and the threshold voltage for the analog trigger channel, call **uTRIGConfigure**.
   - For trigger on rising edge, set mode = **TRIG_LOtoHI**
   - For trigger on falling edge, set mode = **TRIG_HItoLO**

2.12 Miscellaneous Functions

Purpose

This section contains special purpose functions that can be used in a program.

2.12.1 uSetOption Function Calls

Function

uSetOption is used to control the output signal from the UDAS device.

Two options are supported:

- Rate Generator Output
- Analog Output Signals

Rate Generator Output

This option controls the rate generator with output available at the termination panel.

Two rate generator channels are available:

- Analog Input Rate Generator
- Analog Output Rate Generator

A single rate generator output is available at the termination connector. To select one channel to be connected to two outputs, use one **uSetOption** function.

> **Note:** By default, the Analog Input Rate Generator is connected to the output.

Analog Output Signals

On power-up, the signals from the Analog Output channels are disabled. When the Analog Output channels are configured and written, or when a High-Speed Analog Output process is enabled, the output signals are enabled. Use the **uSetOption** function to enable or disable the analog output signals.

2.12.2 Utility Function Calls

Purpose

The host utility functions that do not require communications with the UDAS device include:

- uCountsToTemperature
- uCountsToVolts
- uFrequencyToRGCounts
- uVoltsToCounts

Function

See Appendix A, *U-Link Software Libraries Function Call Reference* for information on each of these utility function calls, including a description and a parameter list.

To convert previously acquired thermocouple and optional cold-junction compensation (CJC) counts values to temperature values, use the **uCountsToTemperature** call.

This function uses the common format counts data for the thermocouple and CJC readings (acquired through **uAIRead** calls), the gain and input ranges used for the readings, and the thermocouple type.

Thermocouple measurements generally require data acquisition from two analog input channels, one to obtain the thermocouple's voltage reading and the other to acquire the CJC data. If a signal conditioning module, such as a PCI-5B37 isolated thermocouple block mounted in a 5B termination panel is used to acquire the thermocouple data, the CJC is already included in the measurement and does not need to be supplied separately.

The **uCountsToTemperature** function supports the conversion and linearization of data acquired from PCI-5B37J, PCI-5B37K and PCI-5B37T non-linearized thermocouple signal conditioning blocks as well as type J, K, and T thermocouples.

To convert the counts value to a floating point voltage value, use the **uCountsToVolts** call. Count values range from -32768 to 32767 for bipolar voltage ranges (such as ±10 V), or from 0 to 65535 for unipolar voltage ranges (such as 0- 10 V). This function is useful for converting data returned from analog input read functions (**uAIRead** and **uAIHSRead**) to voltage counts.

To compute the count value used to program a rate generator channel for the specified frequency use the **uFrequencyToRGCounts** call. The count values are specified to the **uRGConfigure** function to set the rate generator output frequency. If the analog input rate generator is the pacer source for a high-speed analog input process, the frequency programmed is the A/D start convert rate.

> **Note:** If the exact frequency is not attainable, the calculated count values are corrected to achieve the closest frequency possible (less than or equal to the entered frequency).

The UDAS unit's analog input generator supports the frequency range from 0.4 Hz to 4 MHz. The maximum sampling rate of the analog input circuit and the waveform output circuit is 100 kHz.

To convert the specified floating point voltage value to counts, use the **uVoltsToCounts** call. For bipolar voltage ranges (such as ±10 V), the common format count values range from -32768 to 32767. For unipolar voltage ranges (such as 0 - 10 V), the common format count values range from 0 to 65535. This function is useful for converting voltage values to count data for analog output functions that include functions **uAOWrite** and **uTRIGConfigure**.

Appendix A U-Link Software Libraries Function Call Reference

A.1 Introduction

The U-Link Software Libraries provide high-level language support for UDAS-1001E Series Data Acquisition Systems.

Using a Windows operating system, you can link U-Link with Windows application programs written in C/C++, or Microsoft Visual Basic. A common Application Programming Interface (API) is used for all supported operating environments.

A.2 Run-Time Library Reference

All of the U-Link library functions return an integer error code that indicates the success or failure of the function call:

- If the function completes successfully without errors, it returns a 0 value.
- Or, if the function has been completed and one warning is sent to the caller of some potential problem, a negative value warning code is returned.
- If the function did not complete because of some critical error, a positive value error code is returned.

The **I3USBERR.TXT** file included with the U-Link software provides a complete listing of all the error and warning codes that can be returned by the run-time library functions.

Header Files

The various header files (**USBFCNS.H**, **USBDEFS.H**, **I3USBERR.H**) supplied with the library files contain function declarations, constants, and data structure definitions that interface the application program with the functions contained in the libraries.

The programming interface is facilitated through functions contained in the file **I3USB.DLL**. To program in different environments, it is not necessary to learn multiple sets of software calls. Compiler instructions are outlined in Appendix B, *Compiling, Linking, and Running U-Link Applications*.

A.3 Function Calls

This section contains TABLE A.1, *Function Calls* which provides a list of all function calls.

COMMAND	DEFINITION
Initialization	
uDeviceOpen	Opens the UDAS device and performs an initial test.
uDeviceClose	Closes the UDAS handle.
Inquire	
uSYSInquire	Returns information about the connected UDAS unit.
Analog Input	
uAIMAXEnable	Enables use of the AI-MAX expansion box.
uAIConfigureList	Configures a list of analog input channels (such as gain, input range).
uAIRead	Reads a previously configured analog input channel, or all analog channels.
Analog Input (High-Speed)	
uAIHSConfigureList	Configures a high-speed analog input process on a UDAS unit.
uAIHSEnable	Enables or disables a high-speed analog input process.
uAIHSRead	Reads the data acquired in a high-speed analog input process.
uAIHSReturn	Returns data from device buffers to host buffers.
uAIHSStatus	Obtains status and buffer information of a high-speed analog input process.
Analog Output	
uAOConfigureList	Configures a list of analog output channels on a UDAS unit.
uAOWrite	Writes data to a previously configured analog output channel or to all channels.
Analog Output (High-Speed)	
uAOHSConfigureList	Configures a high-speed analog output process on a UDAS unit.
uAOHSEnable	Enables or disables the high-speed analog output process and transmits the data to be output.
uAOHSStatus	Obtains the status of a high-speed analog output process.
uAOHSWrite	Downloads a continued output waveform to the device.
Digital Input/Output	
uDIOConfigureList	Configures a list of digital I/O ports.

TABLE A.1 Function Calls

COMMAND	DEFINITION
uDIORead	Reads one or all I/O ports configured for input.
uDIOWrite	Writes data to one or all output ports.
Digital Input/Output (Bit)	
uDIOBITConfigureList	Configures a list of digital I/O bits.
uDIOBITRead	Reads one or more digital input bits for a specified port.
uDIOBITWrite	Sets the value(s) for digital output. Enables or disables a digital trigger when the bit is configured for one of the trigger modes.
Rate Generator	
uRGConfigure	Configures a rate generator channel on a UDAS unit.
uRGEnable	Enables or disables a rate generator.
Input/Output Option	
uSetOption	Sets optional parameters on a UDAS unit.
Trigger	
uTRIGConfigure	Configures an analog trigger channel for a high-speed analog input or output process.
Counter	
uCTRConfigure	Configures a 16-bit hardware event counter on a UDAS unit for a specified counter mode.
uCTREnable	Starts or stops an event counter, or disables a frequency or time measurement.
uCTRRead	Reads the current count value in event counting modes, or starts a measurement function.
uCTRStatus	Returns the current status of the event-counting or measurement process.
Utilities	
uCountsToTemperature	Converts thermocouple and cold-junction compensation counts values to temperature values.
uCountsToVolts	Converts the specified counts value to a floating point voltage value.
uVoltsToCounts	Converts the specified floating point value to counts.
uFrequencyToRGCounts	Computes count value used to program a rate generator channel for the specified frequency.

TABLE A.1 Function Calls (Continued)

A.4 Function Call Specifications

This section presents the initialization function calls, followed by the I/O functions, then the utility functions. Each category contains a listing of the function calls described in that particular call group. It may be helpful to review the sample programs provided with the U-Link package for detailed example usage of these function calls.

A.4.1 Presentation of Call Specifications

- Call specifications begin with the C/C+ function declaration, followed by a general description of the call.
- References to any other calls that may be interrelated or interdependent with the call being described are included.
- Call parameters are italicized.
- Parameter constant names appear in CAPITAL letters (in the same manner as they are declared in the C/C++ and Visual Basic header file).
- Structure formats are shown in small type, and appear identical to those included the C/C++ and Visual Basic header files.

A.4.2 Supported Software Libraries

The U-Link Software Libraries support:

- Borland C++
- Microsoft Visual C++
- Microsoft Visual Basic

A single dynamic link library, **I3USB.DLL**, is provided for all supported compilers.

Two import libraries are provided:

- **BORIUSB.LIB** for Borland C++
- **MICIUSB.LIB** for Microsoft Visual C++

Separate header and sample program source files are included:

- For C/C++:

  USBFCNS.H contains the run-time library function declarations.

  USBDEFS.H contains constant and type definitions.

All of the application source files that reference the run-time library should include these header files.

- For Visual Basic:

  USBFCNS.BAS contain the run-time library type definitions and function declarations.

  USBDEFS.BAS contain constant definitions.

  These files should be included in the Visual Basic project for the application.

The following files assist the interface with the U-Link drivers and are useful to include in your program: **USBFCNS.BAS**, and **USBDEFS.BAS**.

The file **USBFCNS.BAS** contains the run-time library type and function declarations. The file **USBDEFS.BAS** contains constants and should be included in the Visual Basic project for the application.

A.4.3 Error Codes

All U-Link library functions return either an error code or 0 to the calling program. Always check the return value when a U-Link function is called.

The file **I3USBERR.H** contains error code definitions and descriptions for the error codes that may be generated by the U-Link functions. Some error codes indicate system or bus failures and can be returned by almost any U-Link function. Error codes that are specific to only several functions are listed with each function description in TABLE A.2, *System Error Codes.*

ERROR CODE	DESCRIPTION
INVALID_DEVICE_ERR	The usbHandle parameter of the function is invalid, either out of range or not open.
INVALID_PTR_ERR	A function passed a null pointer for data passed by reference or returned data.
SERIAL_NO_ERR	A packet was received by the host with a non-matching serial number.
HOST_BUFFER_ERR	The host cannot allocate a buffer for a high-speed process.
DATA_LENGTH_ERR	A packet was received by the host with the wrong length.
SEQUENCE_ERR	A response packet was received out of order.
EP3_SEQ_ERR	Bulk packets received out of order on Endpoint 3.
NO_COMMAND_ERR	The command code received by the device is invalid.
BUFFER_ERR	Buffer on the device is not available for a process.
IO_NOT_READY_ERR	The POST test has not completed and the device is not ready to perform I/O functions.
EP1_LENGTH_ERR	The wrong length packet (or no packet) was sent to Endpoint 1.
EP2_LENGTH_ERR	The wrong length packet (or no packet) was received from Endpoint 2.
EP3_LENGTH_ERR	The wrong length packet (or no packet) was sent to Endpoint 3.
EP4_LENGTH_ERR	The wrong length packet (or no packet) was received from Endpoint 4.

TABLE A.2 System Error Codes

A.5 Initialization Functions

uDeviceOpen opens the UDAS device and performs an initial test.

uDeviceClose closes the handle to the UDAS.

uSYSInquire returns information about the connected UDAS unit.

A.5.1 uDeviceOpen

Function

int uDeviceOpen (usbHandle *dev);

Parameters

dev	Returned handle of a UDAS device.

Call Description

This call opens a UDAS device and performs initial tests. This function must be called prior to any other U-Link function call. If this function is called repeatedly, several UDAS devices may be opened simultaneously. The order in which the devices are reported may differ from one program execution to the next, especially if you change the connections. Use the device serial number returned by **uSYSInfo** to identify a specific device.

Returned Error Codes

OUT_OF_DEVICES_ERR	Too many devices open.
DEVICE_OPEN_ERR	Device open failure.
MAILBOX_TEST_ERR	Mailbox pipe response failure.
FIFO_TEST_ERR	FIFO pipe response failure.

A.5.2 uDeviceClose

Function

int uDeviceClose (usbHandle dev);

Parameters

dev	Handle of a UDAS Device to close.

Call Description

This call de-initializes the UDAS unit. This should be the *last* U-Link function call in your program. No command is sent to the UDAS device.

A.5.3 uSYSInquire

Function

int uSYSInquire (usbHandle dev, uSYSInfoType *info);

Parameters

dev	Handle of a UDAS device.
info	System information pointer.

Call Description

This call returns system information about the connected UDAS unit. Use this function to obtain the UDAS board ID, serial number, hardware and firmware versions, and information on UDAS internal errors.

System Information Structure

The following structure is defined in the header file **USBFCNS.H**:

```
typedef struct uSYSInfoType {
    unsigned short id;          // Board ID.
    unsigned long serial_no;    // Serial number.
    short major_version;        // Major ROM version number (hex).
    short minor_version;        // Minor ROM version number (hex).
    short fpga_version;         // FPGA revision.
    short hdwr_version;         // Board revision, containing the FIFO size code.
    unsigned short posterror;   // POST error code.
```

```
unsigned short anlginerr;      // Analog input details.
unsigned short anlgouterr;     // Analog output details.
short bigbufsize:              // Maximum buffer size of firmware.
} uSYSInfoType;
```

The application should call **uSYSInquire** each time it opens or reopens a connection with the UDAS unit and should examine the post error code, since any value other than 0 in this field indicates a hardware problem.

The serial number field is useful for identifying multiple devices attached to a single host.

The `id` indicates the UDAS model:

$0400 -- UDAS multifunction UDAS data acquisition system without analog outputs:
- UDAS-1001E-1 -- regular
- UDAS-1001E-1G -- OEM version
- UDAS-1001E-3 -- with built-in termination

$0401 -- UDAS multifunction UDAS data acquisition system with analog outputs:
- UDAS-1001E-2 -- regular
- UDAS-1001E-2G -- OEM version
- UDAS-1001E-4 -- with built-in termination and analog inputs

The hdwr_version field contains information about the size of the internal FIFOs.

Bits 0-3 indicate the analog input FIFO size; bits 4-7 indicate the analog output FIFO size as follows:

0	no FIFO present
1	512 byte FIFO
2	1024 byte FIFO
3	2048 byte FIFO

A.6 Analog Input Functions

uAIMAXEnable enables the specified termination panels on the AI-MAX 256 channel analog input expansion system.

uAIConfigureList configures a list of analog input channels (such as gain, and input range).

uAIRead reads a previously configured analog input channel or all analog input channels.

uAIHSConfigureList configures a high-speed analog input process on the UDAS unit.

uAIHSEnable enables (starts) or disables (stops) a high-speed analog input process.

uAIHSRead reads the data acquired in a high-speed analog input process and returns information about the data buffer.

uAIHSReturn returns the data acquired in a high-speed analog input process from device buffers to host buffers.

uAIHSStatus obtains the status and current buffer information of a high-speed analog input process.

A.6.1 uAIMAXEnable

Function

int uAIMAXEnable (usbHandle dev, short count, short * panels);

Parameters

dev	Handle of UDAS device.
count	Number of termination panels.
panels	Panel list, index numbers of active panels, up to 15.

Call Description

This call enables the specified termination panels on the AI-MAX 256 channel analog input expansion system. If AI-MAX is used, this function must be called *before* any analog input functions, and before any digital I/O functions that reference Port 1.

The AI-MAX controller uses bits 4 through 7 of digital Port 1. If **AI_MAX** is used, DIOPort 1 may only be configured as output.

Return Error Codes

CONFIGURE_SEQ_ERR	The function was called after an analog configure function, or after a digital configure function that configured bits used by AI-MAX.
AI_PANEL_ERR	Panel number out of range.

A.6.2 uAIConfigureList

Function

int uAIConfigureList (usbHandle dev, short count, uAIListType *list);

Parameters

dev	UDAS device handle.
count	Channel list count. Specify the number of list elements (channels) to configure.
list	Channel list structure pointer.

Call Description

Use **uAIConfigureList** to configure a list of analog input channels with parameters such as gain and input range. You must use this call to configure analog input channels before input data can be read using the **uAIRead** call. Otherwise, the UDAS unit returns an error. If an AI-MAX is connected, use **uAIConfigureList** configure up to 256 analog input channels; otherwise you can configure up to 16 analog input channels.

The **uAIListType** structure contains a panel field which indicates the panel that is being configured. Use the **uAIMAXEnable** call to enable the termination panels that you plan to use. Any combination of channels and parameters may be configured.

Analog Input Channel List Structure

The analog input channel list structure is defined in header file **USBFCNS.H**:

```
typedef struct uAIListType {
    short panel;            // Panel number, 0-15.
    short channel;          // Channel number 0-15.
    short gain;             // Gain: 1, 10, 100.
    short range             // Range code (see below).
    short differential;     // Differential flag (1 or 0).
    short zchannel;         // Zero-reference flag (1 or 0).
    } uAIListType;
```

These are the valid gain codes defined in **USBDEFS.H**:

GAIN_1	Gain of one
GAIN_10	Gain of ten
GAIN_100	Gain of one hundred

The valid range codes, defined in **USBDEFS.H**

UNIPOLAR_10	0-10V
BIPOLAR_10	±5V
BIPOLAR_20	±10V

If the gain specified for a channel is **GAIN_RESET**, the channel will be unconfigured, no longer included in the *all channels* specification. Unconfiguring a differential channel frees two channels for single-ended use. Channels may be unconfigured and reconfigured in the same call. The entry that is being unconfigured must appear first in the list.

Returned Error Codes

AI_PANEL_ERR	Panel number is out of range, or is not specified as active by uAIMAXEnable.
AI_CHANNEL_ERR	Channel number out of range.
AI_DIFF_CONFLICT	Attempt to configure a channel already in use as part of a differential pair.
AI_GAIN_ERR	Invalid gain value.
AI_RANGE_ERR	Invalid range specification.

A.6.3 uAIRead

Function

int uAIRead (usbHandle dev, short panel, short channel, unsigned long count, short *data);

Parameters

dev	Handle of the UDAS device.
panel	Panel number or ALL_PANELS.
channel	Channel number or ALL_CHANNELS.
count	Number of samples per channel to read.
data	Buffer for returned data.

Call Description

This call reads a previously configured analog input channel or all analog input channels. The number of counts per channel can also be specified.

> **Note:** The channels must have been previously configured using **uAIConfigureList**.

Data is returned in *common format* counts format. If a bipolar input range (±5V or ±10 V) is used, the data values will range from -32768 to 32767. If the unipolar input range (0-10 V) is used, the data values will be from 0 to 65535.

To convert common format data to floating point voltage values, use the **uCountsToVolts** function.

The buffer that receives the data (pointed to by the data parameter) must be dimensioned properly if more than one sample is returned or if all channels or all panels are read.

If the code **ALL_CHANNELS** is used, all channels on the panel which have been configured are read.

The code **ALL_PANELS** will cause all panels listed by the **uAIMAXEnable** call to be read. In this case, at least one channel should have been configured on each panel.

If **ALL_CHANNELS** is specified, the returned data array must allocate space for 16 channels, even though fewer channels may be configured.

If **ALL_PANELS** is specified, the returned data array must allocate space for 16 panels with one or 16 channels each (depending on channel specification) even though not all panels are present. The allocated data area must further be increased by multiplying by the number of samples per channel.

If **ALL_CHANNELS** and **ALL_PANELS** are specified and *count* is equal to *n*, the returned data is organized as follows:

panel 0, channel 0, reading 0
panel 0, channel 0, reading 1
…
panel 0, channel 0, reading n-1
panel 0, channel 1, reading 0
panel 0, channel 1, reading 1
…
panel 0, channel 1, reading n-1
…
panel 0, channel 15, reading 0
panel 0, channel 15, reading 1
…
panel 0, channel 15, reading n-1
panel 1, channel 0, reading 0
panel 1, channel 0, reading 1
…
panel 1, channel 0, reading n-1
…
panel 15, channel 15, reading n-1

The parameters specified must not result in return of more than 1021 samples in a single call to **uAIRead**.

Returned Error Codes

AI_PANEL_ERR	Panel number out of range or not specified as active by uAIMAXEnable.
AI_NOT_CONFIGURED	Channel specified has not been configured.
AI_CHANNEL_ERR	Channel number out of range.
AI_DATASIZE_ERR	The channels specified, together with the value of *count* require too large a data buffer.
AI_TIMEOUT_ERR	A time-out failure of the A/D converter was detected.
T2_CONFLICT_ERR	Timer 2 in use by frequency measurement function.

A.7 Analog Input High-Speed Functions

A.7.1 uAIHSConfigureList

Function

int uAIHSConfigureList(usbHandle dev, short start, short stop, short pacer, short trigger, short count, uAIListType *list, short *clustersize);

Parameters

dev	Handle of UDAS device.
start	Initiation mode: start on command (START_COMMAND) or on trigger (START_TRIGGER).
stop	Termination mode: stop on command (STOP_COMMAND), trigger (STOP_TRIGGER), or on terminal count (STOP_COUNT).
pacer	Pacer source, AI Rate Generator (AI_PACER_RGEN) or External Interrupt (AI_PACER_EXT).
trigger	Trigger source, External Interrupt, low to high (AI_TRIG_HI_EXT) or External Interrupt, high to low, (AI_TRIGGER_LO_EXT) or Analog trigger (AI_TRIGGER_AI), rising or falling edge.
count	Channel list count.
list	Channel list structure pointer.
clustersize	Returned cluster size.

Call Description

This call configures a high-speed analog input process on a UDAS unit. A list of the channels and channel parameters and the high-speed parameters are specified to the call to configure the high-speed process.

The channel list uses the same list structure format (**uAIListType**) as the **uAIConfigureList** call described in this section. The **uAIHSConfigureList** high-speed parameters allow specification of start and stop modes, a pacer source and a trigger source.

After configuring all of the elements of the process, the **uAIHSEnable** call is used to enable the high-speed process. **uAIHSEnable** can be used to disable the high-speed process and free the UDAS unit's analog input data buffer after the process completes and the data has been read.

The **uAIHSConfigureList** call returns a value in its clustersize parameter which is the number of bytes required to store the analog input data for one pass through the channel list. The clustersize value can be used to determine the local buffer size required to store the analog input data returned by the **uAIHSRead** function.

Returned Error Codes

AIHS_RUNNING_ERR	A high-speed acquisition is running and should be stopped before configuring a new one.
AI_CHANNEL_ERR	A channel specified is invalid for high-speed data acquisition.
AI_PANEL_ERR	A panel specified is invalid for high-speed data acquisition.
AIHS_LIST_ERR	The channel sequence specified is invalid for high-speed data acquisition.
AIHS_PARAM_ERR	Gain, range, and differential parameters of the channels in the list are not identical.
AIHS_ZCHAN_ERR	The auto-zero parameter is specified for a channel, not allowed for high- speed data acquisition.
AI_GAIN_ERR	Invalid gain value or inconsistent gain values.
AI_RANGE_ERR	Invalid range specification, or inconsistent ranges.
AIHS_PACER_ERR	Invalid pacer specification.
AIHS_TRIGGER_ERR	Invalid trigger specification.
AIHS_START_ERR	Invalid acquisition start condition.
AIHS_STOP_ERR	Invalid acquisition stop condition.

A.7.2 uAIHSEnable

Function

int uAIHSEnable (usbHandle dev, short enable, unsigned long count, unsigned long delay, short clustersize, short free);

Parameters

dev	UDAS device handle.
enable	Enable flag: 1 to enable; 0 to disable.
count	Samples per channel (number of clusters). This is the size of the host buffer allocated.
delay	Trigger delay, in clusters. May not exceed 65535.
clustersize	Cluster size returned by uAIHSConfigureList that is used by the host.
free	Free data buffer memory is used by the host and device if enable = 0.

Call Description

This call is used to enable (start) or disable (stop) a high-speed analog input process.

How it Works

This call starts or stops data acquisition. When data acquisition is started, a data buffer is allocated in the host memory for returned data. The size of the buffer is determined by the *count* parameter.

If the configured start mode is **START_COMMAND**, data acquisition begins immediately. If the configured start mode is **START_TRIGGER**, acquisition does not begin until the trigger condition is met. If the configured stop mode is **STOP_COMMAND**, this function must be called again with *enable* set to 0 to stop the acquisition.

If you specify **STOP_COUNT** as the stop mode in the **uAIHSConfigureList** function, the high-speed analog input process automatically stops when the terminal count is reached (i.e. the specified number of samples per channel have been acquired). On **STOP_TRIGGER**, it will stop when the trigger has occurred and the specified *delay* number of clusters has been acquired.

Depending on the start and stop options chosen in **uAIHSConfigureList** and the value of the *delay*, the number of data clusters acquired may be larger or smaller than the *count* parameter. If the amount of data acquired is greater than the buffer size, the function manages the buffer as a circular buffer.

Once the data acquisition is complete, the data is read beginning with the oldest data. The circular buffer is maintained on the host, not the UDAS device. In order to transfer data to the host, the application makes calls to **uAIHSReturn**. If the buffer space on the device is full, an error occurs.

As data is acquired, buffers will be queued for the **uAIHSReturn** function. The host reads from the device and stores the data in the allocated memory. The **uAIHSRead** command returns data from the host memory to the application.

- If you want to free the UDAS unit's memory used for the data buffer, set the *free* flag when the process is disabled (i.e. when the enable flag is zero).

Returned Error Codes

AIHS_NOT_CONFIGURED	A high-speed acquisition is not configured before enabling.
TRIG_NOT_CONFIGURED	The configured acquisition requires a trigger which has not been configured.
TRIG_CHAN_ERR	The trigger channel is not included in the scan list.
PAC_NOT_CONFIGURED	The configured acquisition requires a pacer which has not been configured.
BUFFER_ERR	Buffer not available for the high-speed process.
AI_RESET_ERR	Hardware error

Modes

If the configured mode is **START_TRIGGER** and **STOP_COUNT**:

- The number of clusters acquired is equal to the *delay* value. For optimum performance, assign identical values for *count* and *delay*.

If the configured mode is **START_COMMAND** and **STOP_TRIGGER**:

- The number of clusters acquired is undetermined, and the buffer may be overwritten several times while waiting for the trigger.
- The maximum number of clusters available is defined in the *count* parameter.
- If the value specified for *delay* is less than the value of the *count* parameter, the final buffer will contain both pre-trigger and post-trigger data; however, if the trigger occurs immediately after the start command, the buffer may not be full. If delay and count are of equal value, the buffer will contain only post-trigger data. If the *count* parameter is less than *delay*, the buffer will contain data starting after the trigger event occurs.

If the configured mode is **START_COMMAND** and **STOP_COUNT**:

- The number of clusters acquired is equal to delay. For optimum performance, the values for *count* and *delay* should be equivalent.

If the configured mode is **START_COMMAND** or **START_TRIGGER** and **STOP_COMMAND**:

- The number of clusters acquired is undetermined and the buffer may be overwritten several times, or only partially filled. The maximum number of clusters available is specified in the *count* parameter, and *delay* is not used for this configuration.

A.7.3 uAIHSRead

Function

int uAIHSRead (usbHandle dev, long offset, long count, short *data, uBufInfoType *info, short *status, long *xcount);

Parameters

dev	Handle of UDAS device.
offset	Cluster offset from beginning of buffer. Parameter can be used to read data starting at a particular cluster. A cluster is a set of input data acquired for one pass through the channel list.
count	Number of samples per channel to read.
data	Returned data buffer pointer. The client data buffer size must be at least (count * clustersize) bytes long.
info	Returned buffer information structure pointer.
status	Returned high-speed analog process status pointer. See the status parameter description in the uAIHSStatus call for more information.
xcount	Returned number of samples per channel that were read. The number of samples returned may be less than the total number of samples requested.

Call Description

This call reads the data acquired in a high-speed analog input process. This call also returns information about the data buffer (*info*) which is described under **uAIHSReturn**.

Returned Error Codes

AIHS_NOT_ENABLED	High-speed acquisition must be enabled before calling this function.
AIHS_CLUSTER_ERR	The specified cluster *offset* is invalid.

A.7.4 uAIHSReturn

Function

int uAIHSReturn (usbHandle dev, uBufInfoType * info, short *status);

Parameters

dev	Handle of UDAS device.
info	Returned buffer information structure pointer.
status	Returned status of the acquisition.

Call Description

This call returns the data acquired in a high-speed analog input process from device buffers to host buffers (transfers the data via USB from the UDAS unit to the host computer). When the data acquisition begins, the host buffer is automatically allocated. Obtaining actual data from the buffer is accomplished by calling **uAIHSRead**.

If the data acquisition will acquire less than 64 Kbytes of data, this function is not needed by the application. The data return function will be performed automatically by **uAIHSRead**. If more than 64 Kbytes will be acquired, **uAIHSReturn** must be called repeatedly during the acquisition because data space on the device is limited. This call returns status and information about the data buffer (*info*). The status bits are described under **uAIHSStatus**.

Buffer Information Structure

The buffer information structure is defined in the header file **USBFCNS.H**:

```
typedef struct uBufInfoType {
    short wrap;                 // Latched buffer wrap flag.
    short running;              // Process running in buffer flag.
    short bytespercluster;      // Number of bytes per cluster.
    short framespercluster;     // Number of frames per cluster.
    long count;                 // Number of clusters available for decoding/encoding.
    long nextcluster            // Index of next cluster to be read.
    long triggercluster;        // Index of trigger cluster.
    } uBufInfoType;
```

The buffer described is the returned data buffer in the host that is managed by the function **uAIHSRead**.

Returned Error Codes

AIHS_NOT_ENABLED	High-speed acquisition must be enabled before calling this function.

A.7.5 uAIHSStatus

Function

int uAIHSStatus (usbHandle dev, uBufInfoType *info, short *status);

Parameters

dev	UDAS device handle.
info	Returned buffer information structure pointer. See the buffer information structure description for more details.
status	Returned high-speed analog process status.

Call Description

This calls returns the status and current buffer information of a high-speed analog input process. In particular, this call is useful for monitoring when a *Stop on Terminal Count* process completes, or when a trigger has occurred.

Use the following library-supplied bit masks to test for particular status conditions.

The status bit codes are defined in the **USBDEFS.H**:

AI_STAT_CONFIGURED	High-speed acquisition has been configured.
AI_STAT_RUNNING	High-speed acquisition is running.
AI_STAT_TRIGGERED	High-speed acquisition has been triggered by analog or external trigger.
AI_STAT_OVERRUN	An overrun of the FIFO has occurred.
AI_STAT_TERMCT	High-speed acquisition has reached terminal count.
AI_STAT_OFFSET	Converter offset correction performed by the host driver.
AI_STAT_DMA_ERR	DMA error (internal)
AI_STAT_BUFF_ERR	Out of buffer space during acquisition.

Returned Error Codes

AIHS_NOT_ENABLED	High-speed acquisition is must be enabled before calling this function.
AIHS_NOT_CONFIGURED	High-speed acquisition is not configured before checking the status.

A.8 Analog Output Functions

uAOConfigureList configures a list of analog output channels on a UDAS unit.

uAOWrite writes data to a previously configured analog output channel, or all channels.

A.8.1 uAOConfigureList

Function

int uAOConfigureList (usbHandle dev, short count, uAOListType *list);

Parameters

dev	Handle of UDAS device acquired through the uDeviceOpen call.
count	Channel list count (1 or 2).
list	Channel list. Channel list elements are of type uAOListType shown below.

Call Description

This call configures a list of analog output channels on the UDAS unit. Analog output channels must be configured using this call before output data can be written to these channels.

Analog Output Channel List Structure

```
typdef uAOListType struct {
    short channel;          // Channel.
    short range;            // Range code: ±10.
    }uAOListType;
```

This function configures analog output channels for single-value outputs.

Returned Error Codes

AO_CHANNEL_ERR	Analog output channel is out of range.
AO_RANGE_ERR	The specified range parameter is invalid.
AO_COUNT_ERR	Number of channels in list is invalid.

A.8.2 uAOWrite

Function

int uAOWrite (usbHandle dev, short channel, short *data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number (0 or 1) to write or **ALL_CHANNELS**.
data	Pointer to the array of counts. Data must be specified in common format counts (-32768 to 32767 for -10 to +10 V). If all channels are written, the data buffer must be in the following format: channel 0, channel 1.

Call Description

This call writes a previously configured analog output channel, or all output channels. The counts data buffer must be dimensioned and initialized properly for the number of channels configured. The **uVoltsToCounts** utility function can be used to convert floating point voltage values to common format D/A counts.

Returned Error Codes

AO_CHANNEL_ERR	Analog output channel is out of range.

A.9 Analog Output High-Speed Functions

uAOHSConfigureList configures a high-speed analog output process on UDAS unit.

uAOHSEnable enables or disables the high-speed analog output process and transmits the data to the output.

uAOHSStatus obtains the status information of a high-speed analog output process.

uAOHSWrite downloads a continued output waveform to the device.

A.9.1 uAOHSConfigureList

Function

int uAOHSConfigureList (usbHandle dev, short pacer, short trigger, short count, uAOListType *list, short *clustersize);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
pacer	Pacer source (required). This is the clock source for hardware D/A update cycles: 0 for Analog Output Rate Generator output (**AO_PACER_RGEN**) 1 for External Input Line (**AO_PACER_EXT**)
trigger	Trigger source, None (**AO_SYNCH_NONE**) or AI synchronous (**AO_SYNCH_AI**).
count	Channel list count. The number of channels in the list.
list	Channel list. Each list element is a type of uAOListType. Refer to the uAOListType structure described in the uAOConfigureList call.
clustersize	Returned cluster size. The clustersize is the number of bytes required to store the output data for one pass through the channel list. This returned information must be passed to the uAOHSEnable and uAOHSWrite functions.

Call Description

This call configures a high-speed analog output process on a UDAS unit. A list of the channels and channel parameters and the high-speed parameters are specified to the call to configure the high-speed process. The channel list uses the same list structure format (**uAOListType**) as the **uAOConfigureList** call described in the section A.7.1, *uAIHSConfigureList*. The **uAOHSConfigureList** high-speed parameters allow specification of a pacer source and a trigger source. After all of the elements of the process are configured, the **uAOHSEnable** call is used to enable the high-speed process and transmit an initial waveform to the output.

Returned Error Codes

AOHS_INUSE_ERR	A high-speed analog output is already running and should be stopped before configuring a new one.
AOHS_LIST_ERR	The channel list sequence is invalid for high-speed analog output.
AO_RANGE_ERR	The specified range parameter is invalid.
AOHS_PACER_ERR	The specified pacer parameter is invalid.
AOHS_SYNCH_ERR	The specified trigger parameter is invalid.

A.9.2 uAOHSEnable

Function

int uAOHSEnable (usbHandle dev, short enable, unsigned long count, short clustersize, short *data, short repeat, short free);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
enable	Enable flag. Set this flag to 1 to enable, or 0 to disable the high-speed analog process.
count	Samples per channel (number of clusters).
clustersize	Cluster size returned by uAOHSConfigureList.
data	Array of output data which contains the waveform to download. To resume output of the loaded waveform, set this pointer to NULL.
repeat	Repeat flag. Set the flag to 1 when the process is enabled (enable flag set to 1) to continuously repeat the output buffer.
free	Free output data buffer (on device). Set the flag to 1 when the process is disabled (enable flag set to zero (0) to free the UDAS unit's data buffer used for this high-speed process. 0 = Do not free the buffer. Set the flag to 0 when the process is enabled.

Call Description

This call enables (start or disables stop) a high-speed analog output process and transmits the data to be output. If the data pointer passed is NULL, the command will resume output of a waveform already transferred and present in the UDAS FIFO. If there is no waveform present, an error **AOHS_DATA_ERR** occurs.

The output waveform is passed in the same format as data for **uAOWrite**. If data is written for both channels, data values for the two channels are interleaved with data for channel 0 given first.

For the best speed performance, the waveform should fit into the analog output FIFO, whose size can be obtained with the **uSYSInfo** function. A waveform shorter than approximately 64 Kbytes can be output as a repeated waveform, stored entirely on the UDAS device. Longer waveforms must be output as non-repeating waveforms using the **uAOHSWrite** call to continue the output. Resources (memory, processor bandwidth, and bus bandwidth) for continued waveforms may be reduced by a simultaneous high-speed analog input process.

Returned Error Codes

AOHS_INUSE_ERR	A high-speed analog output is already running and should be stopped before enabling a new one.
AOHS_CONF_ERR	A high-speed analog output has not been configured.
AOHS_RECV_ERR	A time-out occurred on the device while waiting for the data buffer to be received.
AOHS_DATA_ERR	No data was transmitted and the FIFO was not previously loaded.

A.9.3 uAOHSStatus

Function

int uAOHSStatus (usbHandle dev, unsigned short *status);

Parameters

dev	UDAS device handle.
status	Returned high-speed analog process status.

Call Description

This call obtains the status information of a high-speed analog output process.

The status bit codes are defined in **USBDEFS.H**:

AO_STAT_CONFIGURED	High-speed output has been configured.
AO_STAT_RUNNING	High-speed output is running.
AO_STAT_REPEAT	High-speed output configured for continuous repeat.
AO_STAT_BIGBUFF	Large buffer mode: waveform size exceeds the FIFO length.

AO_STAT_LASTBUFF	The last buffer (non-repeat mode) has been loaded.
AO_STAT_LOADED	A waveform has been loaded into the FIFO.
AO_STAT_EMPTY	FIFO empty.
AO_STAT_UNDERRUN	FIFO empty in non-repeat mode.
AO_STAT_ABORT	High-speed output aborted.
AO_STAT_NOWAIT	OK to transmit next buffer in non-repeat mode.

A.9.4 uAOHSWrite

Function

int uAOWrite (usbHandle dev, unsigned long count, short clustersize, short *data);

Parameters

dev	UDAS device handle acquired through uDeviceOpen call.
count	Samples per channel (number of clusters) to be added to the output waveform from the data input.
clustersize	Cluster size returned by uAOHSConfigureList.
data	Array of output data which contains a waveform segment to download.

Call Description

This call downloads a segment of a continued analog output waveform to the device. For continuous operations, data must be supplied to the device at a rate sufficient to prevent the buffer from being emptied. If the buffer becomes empty, the output terminates.

Return Error Codes

AOHS_NOT_RUNNING_ERR	A high-speed analog output process is not running.
AOHS_WRITE_ERR	The high-speed analog output process is configured to automatically repeat the FIFO data.
AOHS_RECV_ERR	A time-out occurred on the device while waiting for the data from the buffer to be received.

A.10 Counter Functions

uCTRConfigure is used to configure a 16-bit hardware counter on the UDAS unit for specified counter mode.

uCTREnable is used to start or stop an event counter, or to disable a frequency or time measurement.

uCTRRead reads the current count value in event-counter mode, or starts a measurement.

uCTRStatus returns the current status of event-counting or measurement process.

A.10.1 uCTRConfigure

Function

int uCTRConfigure (usbHandle dev, short channel, short mode, unsigned long gate);

Parameters

dev	UDAS device handle acquired through uDeviceOpen.
channel	Channel number. Parameter should always be zero.
mode	Counter mode. See the list of valid counter modes provided below.
gate	Gate interval for frequency measurement. See the recommended values given below.

Call Description

This call configures the 16-bit hardware counter on the UDAS unit for the specified counter mode. If the counter has already been configured and enabled, it must be disabled before it can be reconfigured. The counter is implemented in software to provide a 32-bit count.

Valid Counter Modes

Valid counter modes are defined in **USBDEFS.H**:

CTR_MODE_EVENT_HI	Event counter, gate high to count.
CTR_MODE_EVENT_LO	Event counter, gate low to count.
CTR_MODE_FREQ	Frequency measurement.
CTR_MODE_PERIOD	Period measurement.
CTR_MODE_PULSE_HI	High pulse measurement.

CTR_MODE_PULSE_LO	Low pulse measurement.

Event Counter Modes

In these modes, the counter acts as a 32-bit counter. It counts when the gate is true, stops when the gate is false. The count read by **uCTRRead** may be cumulative or incremental. If your application does not use the gate, connect it to ground and use **CTR_MODE_EVENT_LO**.

Frequency Measurement

For frequency measurement, the following gate values are recommended in **USBDEFS.H**:

FRQ_GATE_1ms	1ms gate
FRQ_GATE_10ms	10 ms gate
FRQ_GATE_100ms	100 ms gate
FRQ_GATE_1s	1s gate

The gate is implemented internally and the external gate input is not used. The precision of the measurement is determined by the gate value used.

Period and Pulse Measurements

In these modes, the counter is internally connected to an 8 MHz reference signal. The counter counts the reference signal for the duration of the desired period.

Returned Error Codes

CTR_CHANNEL_ERR	Channel number specified must be zero.
CTR_INVALID_MODE	Counter mode specified is invalid.
CTR_ENABLE_ERR	Counter cannot be configured while enabled.
CTR_GATE_ERR	Gate specified for frequency measurement is invalid.

A.10.2 uCTREnable

Function

int uCTREnable (usbHandle dev, short channel, short enable);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number. Should be set to 0.
enable	Enable flag: Use any non-zero value to enable, 0 to disable.

Call Description

This call starts or stops an event counter, or disables a frequency or time measurement process. In event counter mode, the enable function initiates counting and the disable function stops counting. In time measurement modes, (frequency, period, and pulse measurement), the **uCTREnable** function does not need to be called before the **uCTRRead** function.

The **uCTREnable** function with a disable flags halts the measurement in progress, and disables the interrupts. The counter must be disabled in any mode before being reconfigured.

Returned Error Codes

CTR_CHANNEL_ERR	Channel number specified must be zero.
CTR_NOT_CONFIGURED	Counter not configured before enabling.
CTR_READ_PROG_ERR	Measurement was in progress when the counter was disabled.

A.10.3 uCTRRead

Function

int uCTRRead (usbHandle dev, short channel, short reset, unsigned long *data, unsigned short *status);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number. Should be set to zero.
reset	Reset count flag.
data	Returned data. Frequency readings are returned in Hertz; event counter readings are returned in counts; period and pulse width measurements are returned in nanoseconds.
status	Returned status.

Call Description

In event counter mode, this call reads the current count value. If the reset parameter is true, the counter is reset, and the next reading will return an incremental count rather than a cumulative count.

In measurement modes, the first time this function is called it will start the measurement. If the measurement completes immediately, the data is returned. Otherwise, additional calls must be made until the *status* indicates that the measurement is complete and data is returned. If the *reset* parameter is true, a new measurement is started as soon as the current one completes.

Note that the frequency measurement function uses an internal timer resource which is also required by the **uAIRead** function. If frequency measurements are intermixed with analog readings, there may be a resource conflict resulting in an error.

The following status bits may be returned:

CT_STAT_MIPROG	Measurement in progress (period, pulse, or frequency measurements).
CT_STAT_RUNNING	Counter running.
CT_STAT_NOTCONF	Counter not configured.

If the returned status indicates **CT_STAT_MIPROG**, the data value is not valid. The host must make additional calls to **uCTRRead** or **uCTRStatus** until the reading is completed.

Return Error Codes

CTR_CHANNEL_ERR	Channel number specified, must be zero.
T2_CONFLICT_ERR	Timer 2 in use by analog input function.
CTR_NOT_CONFIGURED	Counter not configured before reading.
CTR_LATCH_ERR	Counter latch failed repeatedly.

A.10.4 uCTRStatus

Function

intuCTRStatus (usbHandle dev, short channel, unsigned short *data, unsigned short *status);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number. Should be 0.
status	Returned status.

Call Description

This returns the current status of the event-counting or measurement process.

The following status bits may be returned:

CT_STAT_MIPROG	Measurement in progress (period, pulse, or frequency measurements).
CT_STAT_RUNNING	Counter running.
CT_STAT_NOTCONF	Counter not configured.

Returned Error Codes

CTR_CHANNEL_ERR	Channel number specified must be 0.
CTR_NOT_CONFIGURED	Counter not configured before checking the status.

A.11 Digital Input/Output (Port) Functions

uDIOConfigureList configures a list of digital I/O ports.

uDIORead reads one or two digital I/O ports configured for input.

uDIOWrite writes data to one or two digital output ports.

A.11.1 uDIOConfigureList

Function

int uDIOConfigureList (usbHandle dev, short count, uDIOListType *list);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
count	Number of ports to configure, 1 or 2.
list	Port list. Port list elements are of type uDIOListType.

Call Description

This call configures a list of digital I/O ports. Digital I/O ports must be configured using this call before input or output data can be read from or written to these ports.

Digital I/O Port List Structure

The digital I/O ports list structure is defined in the header file **USBDEFS.H**:

```
typedef struct uDIOListType {
    short port;          // Port number.
    short mode;          // DIO mode, described below.
    short data;          // Initial output data.
    } uDIOListType;
```

The supported DIO modes are defined in the header file **USBDEFS.H**:

DIO_MODE_INPUT	Input port.
DIO_MODE_OUTPUT	Output port.

This call may be used to configure as inputs those ports whose bits will be configured as trigger inputs. It may be used to reconfigure as inputs or outputs those ports whose bits have been configured otherwise.

An AI-MAX controller uses bits 4 through 7 of Port 1. If AI-MAX is used, Port 1 may only be configured as an output, and writes to Port 1 only affect bits 0 through 3. **uAIMAXEnable** should be called before any digital configuration calls.

Return Error Codes

DIO_PORT_ERR	The specified port value is invalid.
AIMAX_CONFLICT_ERR	Port 1 is being configured in conflict with its use by AI-MAX.
DIO_MODE_ERR	The specified mode value is invalid.

A.11.2 uDIORead

Function

int uDIORead (usbHandle dev, short port, unsigned short *data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
port	Port number, or ALL_PORTS.
data	Data array. If all ports are read, the data array will contain 2 elements.

Call Description

This call allows you to read one or two digital I/O ports configured for input. Specify the port or ports that you want to read in the call's *port* parameter. Specify a port number, or all ports (**ALL_PORTS**).

If the port is configured as an output, this function will read back the output registers. If individual bits are configured as trigger input bits, their current state will be returned by this call.

The data buffer must be dimensioned properly for the number of ports to be read.

Returned Error Codes

DIO_PORT_ERR	The specified port value is invalid.
DIO_MODE_ERR	The specified port is configured in an illegal mode for this function.
DIO_NOT_CONFIGURED	The specified port has not been configured at all.

A.11.3 uDIOWrite

Function

int uDIOWrite (usbHandle dev, short port, unsigned short *data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
port	Port number, or **ALL_PORTS**.
data	Data array. If all ports are read, the buffer consists of two elements, containing the data for ports 0 and 1.

Call Description

This call writes data to one or two digital output ports. The port(s) must be configured for output operation, otherwise, an error will result. Specify the port or ports that you want to read in the call's port parameter. You can specify a port number or all ports (**ALL_PORTS**). The data buffer must be dimensioned and initialized properly for the number of ports.

Returned Error Codes

DIO_PORT_ERR	The specified port value is invalid.
DIO_MODE_ERR	The specified port is configured in an illegal mode for this function.
DIO_NOT_CONFIGURED	The specified port has not been configured at all.

A.12 Digital Input/Output (Bit) Functions

A.12.1 uDIOBITConfigureList

Functions

int uDIOBITConfigureList (usbHandle dev, short count, uDIOBITListType *list);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
count	List count. Number of bits to be configured. A UDAS unit has two 8-bit ports (16 bits).
list	Bit list. Bit list elements are of type uDIOBITListType.

Call Description

This call configures a list of digital I/O bits. All digital I/O bits must be configured using this call before input or output data can be read from or written to these bits.

Digital I/O Port Bit List Structure

The digital I/O bit list structure is defined in the header file **USBFCNS.H**:

```
typedef struct uDIOBITListType {
    short port;          // Port number.
    short portbit;       // Bit within the port, or bit mask.
    short mode;          // I/O mode described below.
    short data;          // Initial output data or trigger value.
    } uDIOBITListType;
```

Supported DIO Modes

Supported digital input and output modes are defined in the header file **USBDEFS.H**.

DIO_MODE_INPUT	Input bit.
DIO_MODE_OUTPUT	Output bit.
DIO_MODE_HItoLO	Trigger input bit, high to low trigger.
DIO_MODE_LOtoHI	Trigger input bit, low to high trigger.
DIO_MODE_ANY	Trigger input bit, any edge trigger.

Input and Output Modes

These modes are equivalent to the digital I/O port modes, except that read and write functions communicate data with one data word for each bit, rather than packing the data into a single word.

Trigger Input Modes

Bits configured as trigger inputs generate alarms when the input signal makes a high-to-low, low-to-high transition, or both. The number of such transitions is counted and may be read back as an integer value using **uDIOBitRead**.

Trigger inputs must be armed by calling **uDIOBitWrite** before they will begin counting. The count may be optionally reset when read. These inputs may be used to latch an alarm condition, or to implement low-speed counters. To read the current state of a trigger input bit, read the port with **uDIORead** and unpack the bits.

Trigger inputs are not intended to perform high-speed counting functions. If the UDAS system detects that its resources are overloaded by a trigger input, it will be shut down and an error will be returned the next time the port is referenced.

All bits in one port must be configured either as inputs or outputs, but ordinary inputs and trigger inputs may be mixed. The initial data value (0 or 1) is written to output ports immediately after they are configured as outputs. The data value is ignored for inputs.

If an entry in the list array contains a portbit code specifying more than one bit, all bits specified are configured identically, with identical data parameters.

AI_MAX uses bits 4 through 7 of Port 1. If AI-MAX is used, Port 1 may only be configured for output, and only bits 0 through 3 may be configured. **uAIMAXEnable** should be called before any digital configuration calls.

If for some reason you would like to reconfigure bits from input to output or vice versa, you must call **uDIOConfigureList** to reconfigure the port before you call **uDIOBITConfigureList** to configure the individual bits.

Returned Error Codes

DIO_PORT_ERR	The specified port value is invalid.
DIOB_BIT_ERR	The specified bit number is invalid.
AIMAX_CONFLICT_ERR	Port 1 is being configured in conflict with its use by AI-MAX.
DIO_MODE_ERR	The port has been configured in a mode incompatible with the mode specified for the bit being reconfigured.
DIOB_ENABLE_ERR	A specified bit has been enabled. It must be disabled before it can be reconfigured.
OVERLOAD_ERR	Input signal on a trigger input is too fast.

A.12.2 uDIOBITRead

Function

int uDIOBITRead (usbHandle dev, short port, short portbit, short reset, long *data, unsigned short *status);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
port	Port containing the input bits.
portbit	Bit number, all bits (**ALL_PORT_BITS**), or specify a bit mask to read selected ports.
reset	Reset flag. For trigger input modes, setting this flag to 1 will reset all the counts to 0 after reading. The rest flag is not used in other modes.
data	Data array pointer. If all bits or multiple bits are read (using a bit mask), the data buffer must be dimensioned for the 8 bits (bits 0-7 of the port).
status	Returned status. This is a bit mask indicating with 1 those bits that are configured as trigger inputs and are currently armed.

Call Description

This call is a general purpose digital input bit read function. This call reads a digital I/O bit, selected bits or all bits on a specified port, configured for either **DIO_MODE_INPUT**, or for one of the three trigger input modes, **DIO_MODE_HItoLO**, **DIO_MODE_LOtoHI**, or **DIO_MODE_ANY**.

> **Note:** If the bit is configured for trigger input modes, the uDIOBITWrite function must be called to arm the trigger. See the uDIOBITWrite description for more information.

Specifying a Bit Mask

A bit mask uses the upper 8 bits of the 16-bit *portbit* number to specify which bits to read. For example, hexadecimal number $0900 = 0000 1001 0000 0000 - reads bits 0 and 3. Note the lower 8 mask bits must always be 0 for a bit mask. When reading a single bit by specifying the bit number, the high eight bits must be zero. For example, to read bit 2, set portbit to 2).

If a bit mask is used in the portbit parameter, data will be returned in the corresponding array element(s). For example, if the bit mask $0900 = 0000 1010 0000 0000 is specified, bit 0 and 3 data will be returned in the first and fourth array elements, respectively.

For normal input data, the least significant bit of returned data value (1 or 0) indicates the bit's input state.

For trigger input data, the current trigger count is returned.

Returned Error Codes

DIO_PORT_ERR	The specified port value is invalid.
DIO_BIT_ERR	The specified bit number is invalid.
DIO_NOT_CONFIGURED	The specified port has not been configured at all.
OVERLOAD_ERR	Input signal on a trigger input is too fast

A.12.3 uDIOBITWrite

Function

int uDIOBITWrite (usbHandle dev, short port, short portbit, unsigned short *data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
port	Port containing the output bit(s).
portbit	Bit number, **ALL_PORT_BITS**, or specify a bit mask to set the values of selected bits.
data	Data array pointer. This array contains the value(s) to output. If all port bits or multiple port bits will be written to (using a port bit mask), the data array must contain 8 elements (bits 0-7 of the port).

Call Description

This call may be used to perform two different functions. If the digital bits are configured as output bits, this function sets the output level of the bits.

If the digital bits are configured as trigger input bits, this function arms or disarms the trigger. The data value 1 arms the trigger, the value 0 disarms the trigger. This call is not valid for bits configured as ordinary inputs.

Specifying a Bit Mask

A bit mask uses the upper 8 bits of the 16-bit portbit number to specify which bits to write to. For example, the hexadecimal number $0900 = 0000 1001 0000 0000 - writes to bits 0 and 3. Note the lower 8 mask bits must always be zero.

If a bit mask is used in the portbit parameter, bit data must be stored in the corresponding array element(s). For example, if the port bit mask $0900 = 0000 1001 0000 0000 is specified, data from the first and fourth buffer elements will be written to bits 0 and 3.

Returned Error Codes

DIO_PORT_ERR	The specified port value is invalid.
DIO_BIT_ERR	The specified bit number is invalid.
DIO_MODE_ERR	The specified bit is configured in an invalid mode for this function.
DIO_NOT_CONFIGURED	The specified port has not been configured at all.
OVERLOAD_ERR	Input signal on a trigger input is too fast.

A.13 Rate Generator Functions

uRGConfigure: Configures the analog rate generator channel on a UDAS unit with a count value stored in the buffer pointed to by *data*.

uRGEnable: This function enables or disables a rate generator channel.

A.13.1 uRGConfigure

Functions

int uRGConfigure (usbHandle dev, short channel, unsigned long data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number. Supported channels are AI_RG and AO_RG.
data	24-bit rate generator data.

Call Description

Configures the analog input or analog output rate generator channel on a UDAS unit with a *count* value stored in the buffer pointed to by *data*. Pulses generated by the analog input rate generator are used to start A/D conversions in high-speed analog input processes and update analog output channels, respectively.

A rate generator's output, selected by the **uSETOption** call, is also available at the I/O connector on the UDAS unit. A rate generator's frequency is the result of dividing the on-board 8 MHz timebase frequency by a 24-bit integer count.

The output frequency is given by the formula:

F = 8 MHz/count

where count can range from 2 to 16777215. From the above formula, the output frequency can range from a maximum of 4 MHz to a minimum of 0.476 Hz. The maximum sampling rate supported by the analog input and analog output circuits is 100 kHz.

The **uFrequencyToRGCounts** utility function can be used to obtain the count value for the desired output frequency. The rate generator is disabled by the **uRGConfigure** call prior to configuration or reconfiguration. The **uRGEnable** call, with its enable flag set to zero, may also be used to enable or disable the rate generator.

Returned Error Codes

RG_COUNT_ERR	The rate generator count data can not exceed 24 bits.
RI_CHANNEL_ERR	The rate generator channel specified is invalid.

A.13.2 uRGEnable

Function

int uRGEnable (usbHandle dev, short channel, short enable);

Parameter Descriptions

dev	UDAS device handle acquired through the uDeviceOpen call.
channel	Channel number. Supported channels are AI_RG and AO_RG.
enable	Enable flag. Set this flag to a non-zero value to enable to disable the rate generator. Set this flag to zero to disable the rate generator.

Call Description

This function enables or disables a rate generator channel.

> **Note:** The analog output rate generator is also enabled when the **uAOHSEnable** function (with its enable flag set to 1) is called to start a high-speed analog output process. When using a high-speed analog input process, you must always call uRGEnable before you start the data acquisition process.

Returned Error Codes

RG_NOT_CONFIGURED	The rate generator has not been configured.
RG_CHANNEL_ERR	The rate generator channel specified is invalid.

A.14 I/O Option Function

uSetOption: This call sets the optional parameters on the UDAS unit.

A.14.1 uSetOption

Function

int uSetOption (usbHandle dev, short option, short data, short reserved);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
option	Option code.
data	Option-dependent data.
reserved	Reserved.

The following options are currently supported:

OPTION_RG_OUTPUT	The value of the data for this option is the rate generator channel number, AI_RG or AO_RG. This option specifies the rate generator output which appears at the termination output.
OPTION_AO_DISABLE	The data parameter is not used. This option disables the analog outputs, restoring them to their power-on condition.

Returned Error Codes

OPTION_ERR	Unsupported option code.
RG_CHANNEL_ERR	Incorrect rate generator channel.
AOHS_INUSE_ERR	Attempt to disable analog outputs while high-speed output is running.

A.15 Trigger Function

uTRIGConfigure: Configures an analog input channel as a trigger for a high-speed analog input.

A.15.1 uTRIGConfigure

Function

int uTRIGConfigure (usbHandle dev, short panel, short channel, short mode, short data);

Parameters

dev	UDAS device handle acquired through the uDeviceOpen call.
panel	Analog input panel number. Should be zero if AI-MAX is not used.
channel	Analog input channel number.
mode	Trigger modes (valid trigger modes defined in USBDEFS.H): TRIG_HItoLO Trigger on high-to-low transition TRIG_LOtoHI Trigger low to high transition
data	Threshold counts data. The count value must be in common analog format - integer values from -32768 to 32767 are equivalent to a range of ±10 Volts. To convert floating point voltage values to count values, the uVoltsToCounts utility function can be used.

Call Description

This function configures an analog input channel as a trigger for a high-speed analog input or output process on a UDAS unit. The analog trigger circuit on a UDAS unit uses any analog input channel as an input to an internal comparator.

Analog output channel 1 is used to generate a threshold voltage for the comparator circuit. When the input voltage on the selected input channel crosses the threshold voltage, a trigger signal is generated. When the analog trigger is used, analog output channel 1 should not be used for other purposes.

The analog trigger circuit can be programmed to activate when the input signal rises above or falls below the threshold voltage. Since the analog output circuit has a range of ±10 V, the trigger signal connected to the selected analog input channel must also be within this range.

> **Note:** Any gain programmed for the trigger channel is applied before comparing it with the threshold.

For the analog trigger to be operational for high-speed analog input, the trigger channel must be included in the list of channels scanned. When the high-speed *start on trigger* mode is used, the first channel reading that will be stored is from the trigger channel.

Returned Error Codes

AI_PANEL_ERR	Panel number is out of range or not is specified as active by uAIMAXEnable.
AI_CHANNEL_ERR	Channel number our of range.
TRIG_MODE_ERR	The specified trigger mode is invalid.

A.16 Utility Functions

uCountsToTemperature converts and linearizes previously acquired thermocouples and cold-junction compensation (CJC) counts values to temperature values.

uCountsToVolts converts the specified counts value to a floating point voltage value.

uFrequencyToRGCounts computes the count value used to program a rate generator channel for the specified frequency.

uVoltsToCounts converts the specified counts value to a floating point voltage value.

> **Note:** These are host functions that do not require any communication with the UDAS device.

A.16.1 uCountsToTemperature

Function

int uCountsToTemperature (short tccounts, short tcgain, short tcrange, short cjc, short cjccounts, short cjcgain, short cjcrange, short type, double *temperature);

Parameters

tccounts	Counts read from the thermocouple channel.
tcgain	Thermocouple channel gain value.
tcrange	Thermocouple range code.
cjc	CJC flag: set to one to perform CJC, or set to zero to omit CJC.
cjccounts	Counts read from CJC channel.

cjcgain	Gain value for the CJC channel.
cjcrange	Range code for the CJC channel.
type	Thermocouple type code.
temperature	Returned temperature value.

Call Description

This utility call converts and linearizes previously acquired thermocouple and cold-junction compensation (CJC) counts values to temperature values. **uCountsToTemperature** uses the counts data for the thermocouples and CJC readings (acquired through **uAIRead** calls for example), the gain and input ranges, and the thermocouples type (type J, K or T) to calculate the thermocouple junction's temperature in °C.

To simplify the hook-up and measurement of thermocouples, signal termination panels with built-in CJC sensor channels are available from Intelligent Instrumentation.

In most instances, thermocouple measurement requires data acquisition from two analog input channels, one to obtain the thermocouple's output voltage and the other to acquire CJC data. In addition to standard thermocouple data conversion, **uCountsToTemperature** supports conversion and linearization of data acquired from PCI-5BG37J, PCI-5B37K and PCI-5B37T non-linearized thermocouples signal conditioning blocks.

If the thermocouple's data was acquired using a signal conditioning module, such as a PCI-5B37 isolated thermocouples block mounted in a 5B termination panel, CJC is already included in the measurement and does not need to be separately supplied.

Supported thermocouple types include:

TC_TYPE_J	Type J
TC_TYPE_K	Type K
TC_TYPE_T	TypeT
TC_TYPE_J_5B	Type J (non-linearized 5B)
TC_TYPE_K_5B	Type K (non-linearized 5B)
TC_TYPE_T_5B	Type T (non-linearized 5B)

A.16.2 uCountsToVolts

Function

int uCountsToVolts (short counts, short gain, short range, double *volts);

Parameters

counts	Counts read from an analog channel.
gain	Gain value for the analog channel.
range	Range code for the analog channel.
volts	Returned voltage value.

Call Description

The **uVoltsToCounts** function is useful for converting data returned from analog input read functions **uAIRead** and **uAIHSRead** to voltage units. Count values range from -32768 to 32767 for bipolar voltage ranges (e.g. ±10 V), or from 0 to 65535 for unipolar voltage ranges (e.g. 0 to 10 V).

A.16.3 uFrequencyToRGCounts

Function

int uFrequencyToRGCounts (double frequency, unsigned *count)

Parameters

frequency	Frequency in Hertz.
count	Returned count value for use in the uRGConfigure call.

Call Description

This utility call computes the count value used to program a rate generator channel for the specified frequency. The count values are specified to the **uRGConfigure** function to set the rate generator output frequency. If the analog input rate generator is the pacer source for a high-speed analog input process, the output frequency is the A/D start convert rate.

> **Note:** If the exact frequency is not attainable, the calculated count value is corrected to achieve the closest frequency possible (less than or equal to the entered frequency).

A.16.4 uVoltsToCounts

Function

int uVoltsToCounts (double volts, short range, short *counts);

Parameters

volts	Voltage to output to analog channel.
range	Range code for analog channel.
counts	Returned counts value.

Call Description

The **uVoltsToCounts** function is useful for converting voltage values to count data for analog output functions such as **uAOWrite** and **uTRIGConfigure**.

Count values range from -32768 to 32767 for bipolar voltage ranges (e.g. ±10 V), or from 0 to 65535 for unipolar voltage ranges (e.g.0 - 10 V).

Appendix B Compiling, Linking, and Running U-Link Applications

B.1 General Information

The U-Link Software Libraries provide high-level language support for UDAS data acquisition systems. This version of U-Link software supports the following compilers:

- Borland C++ versions 4.5 and 5.0
- Microsoft Visual C++ versions 5.0
- Microsoft Visual Basic versions 5.0

Microsoft Windows 98 is the operating system for UDAS hardware using the U-Link software.

B.2 Compiling and Linking

Compiling U-Link programs using Borland C++ version 5.0

Include both of the **USBDEFS.H** and **USBFNCS.H** files in all C source files that call functions in the U-Link API. Use the **BORIUSB.LIB** file in your project.

Compiling U-Link programs using Microsoft C++ version 5.0

Include both of the **USBDEFS.H** and **USBFNCS.H** in all C source files which call functions in the U-Link API. Use the **MICIUSB.LIB** file in your project.

Compiling U-Link programs using Microsoft Visual Basic version 5.0

Include the **USBDEFS.BAS**, and **USBFNCS.BAS** files in your project. If you plan to run your Visual Basic under Interactive Development Environment (IDE), temporarily locate the **I3USB.DLL** in the VB directory or anywhere in the search path.

B.3 Running

To execute any of the U-Link functions, your operating system must locate the **I3USB.DLL** file. The location of this important file must be either in the default directory, or in a directory in the search path.

Programs that make calls to the **uCountsToTemperature** function will return error code 79, if your program is searching for a thermocouple table file that it is unable to locate. To correct this problem, copy the proper thermocouple table file to the default directory.

Appendix C Error Code Information

C.1 Successful Operation Code

Success Code	Description	Action
0 (NO_USB_ERROR)	No error.	Operation was successful.

C.2 Error Codes Returned by UDAS Driver DLL

Error Code	Description	Action
1 (INVALID_DEVICE_ERR)	An invalid device handle was passed by the block to the UDAS drivers.	Contact the factory.
2 (INVALID_PIPE_ERR)	An internal error occurred in the UDAS drivers.	Contact the factory.
3 (OUT_OF_DEVICES_ERR)	Too many devices have been opened.	The maximum number of devices open at once is 8.
4 (DEVICE_OPEN_ERR)	The UDAS could not be opened. This might occur if another application, such as another Visual Designer diagram or SYSCHECK, is currently communicating with the device.	Do not run more than one application that accesses the same UDAS device at once. This error could occur if an application terminated without closing the UDAS. In this case, it may be necessary to disconnect and reconnect the USB cable.
5 (MAILBOX_TEST_ERR)	Internal error. There may be a problem with the hardware.	Contact the factory.
6 (FIFO_TEST_ERR)	Internal error. There may be a problem with the hardware.	Contact the factory.
7 (INVALID_PTR_ERR)	Internal error. A Visual Designer block passed an invalid value to the UDAS drivers.	Contact the factory.

Error Code	Description	Action
8 (SERIAL_NO_ERR)	Internal error. The UDAS unit responded to a message improperly. The hardware may be failing.	Contact the factory.
9 (HOST_BUFFER_ERR)	The specified high-speed analog input buffer size is too large to be allocated by the UDAS drivers.	Alter the high-speed analog input settings to reduce the buffer size.

C.3 Error Codes Returned by the UDAS Driver DLL

Error Code	Description	Action
10 (EP1_LENGTH_ERR)	Internal error. The computer sent the unit a packet of an incorrect length.	Contact the factory.
11 (EP2_LENGTH_ERR)	Internal error. The computer received a packet containing an incorrect length from the UDAS unit.	Contact the factory.
12 (EP3_LENGTH_ERR)	Internal error. The computer sent the unit a packet of an incorrect length.	Contact the factory.
13 (EP4_LENGTH_ERR)	Internal error. The computer received a packet from the UDAS which was of the incorrect length.	Although this infrequently occurs, the cause is an unrecoverable bus error.
14 (HOST_MEMORY_ERR)	A host buffer can not be allocated. High-speed acquisitions of more than 64K of data require that the application call uAIHSReturn during the acquisition.	The size of a high-speed analog input buffer may need to be reduced.
19 (EP3_SEQ_ERR)	A packet expected as part of a message to the UDAS device was lost. This error should not occur.	Contact the factory.
20 (NO_COMMAND_ERR)	Invalid command code received by UDAS unit. This could be caused by using a later version of the UDAS drivers with a UDAS unit which does not support one or more of the more advanced driver functions.	Contact the factory.
21 (EE_ADDRESS_ERR)	Internal error in EEPROM addressing.	Contact the factory.
22 (BUFFER_ERR)	There are no available buffers on the UDAS for high-speed analog input or analog output data.	Try reducing the rate at which data is acquired with the analog input process.

C.4 Multi-function Error Codes

Error Code	Description	Action
23 (AIMAX_CONFLICT_ERR)	An attempt was made to configure digital I/O port 1 for input while AI-MAX was in use for analog input channel expansion.	Refer to the hardware manual for restrictions on digital I/O configurations when AI-MAX is in use.
24 (T2_CONFLICT_ERR)	An internal timer on the UDAS unit is used both for analog input and for counter frequency measurement.	An attempt has been made to perform both types of processes simultaneously.

C.5 Digital I/O Error Codes

Error Code	Description	Action
25 (DIO_PORT_ERR)	The specified digital port number is invalid.	Ports 0 and 1 are supported.
26 (DIO_MODE_ERR)	The digital I/O mode specified for the port is not supported by this port. This error can be generated if a digital mode is set for a port which does not support bits configured for that mode.	Check the hardware manual for mode restrictions.
27 (DIO_NOT_CONFIGURED)	An attempt was made to enable, write to, or read from a port which has not been configured.	Verify that the port has been configured before being used.
28 (DIOB_BIT_ERR)	The specified digital I/O bit number is not valid.	Valid bit numbers are 0-7.
29 (DIOB_MODE_ERR)	The specified digital I/O bit mode is not valid for the selected port/bit. This could be caused by attempting to configure one bit in an output port for an input mode, a bit in an input port for an output mode, or a bit in a port for a mode not supported by bits in that port.	Change port direction by reconfiguring the entire port, then configure each bit. Refer to the hardware manual for restrictions on mode assignments.
30 (DIOB_ENABLE_ERR)	An attempt to reconfigure a digital I/O bit was made while the bit was enabled in a trigger mode.	The bit must first be disabled.

C.6 Analog Input Error Codes

Error Code	Description	Action
31 (AI_PANEL_ERR)	The AI-MAX panel selected is not valid (valid range is 0-15), or AI-MAX has not been enabled.	Enable AI-MAX. Verify the AI-MAX panel selected is valid (0-15) and specified by uAIMAXEnable. If AI-MAX is not being used, verify that a panel other than 0 has not been selected.
32 (CONFIGURE_SEQ_ERR)	AI-MAX was enabled on the UDAS unit after the digital output port used to control AI-MAX channel selection had already been configured for an incompatible mode or analog input was configured prior to enabling AI-MAX.	Refer to the hardware manual for information on configuration restrictions on digital I/O ports when AI-MAX is in use.
33 (AI_DIFF_CONFLICT)	An attempt was made to read either the main signal or return signal half of a differential channel as a single-ended channel. This is not supported.	Ensure the application consistently configures and reads the analog input channels.
34 (AI_GAIN_ERR)	An invalid or inconsistent gain value has been specified.	Valid gain values are 1, 10, and 100.
35 (AI_RANGE_ERR)	An invalid or inconsistent range value has been specified.	Valid ranges are 0-10, ±5, and ±10.
36 (AI_NOT_CONFIGURED)	The specified analog input channel was read prior to its configuration.	Call uAIConfigureList first.
37 (AI_CHANNEL_ERR)	An invalid or inconsistent channel number was specified.	Verify the channel number is correct, given the single-ended/differential configuration.
38 (AI_TIMEOUT_ERR)	A time-out occurred while waiting for an analog input conversion to complete. The hardware may be failing.	Contact the factory.
39 (AI_DATASIZE_ERR)	The number of channels and AI-MAX panels specified is too large to acquire the desired number of samples from each channel.	Reduce the number of samples or the number of channels.

C.7 Analog Output Error Codes

Error Codes	Description	Action
40 (AO_CHANNEL _ERR)	An invalid analog output channel number was specified. The UDAS unit may not have analog output channels.	Channels 0 and 1 are supported.
41 (AO_RANGE_ERR)	An invalid range parameter was passed for configuration of an analog output channel.	Valid range is ±10.
42 (AO_COUNT_ERR)	An invalid number of channels were specified in the configuration list for the analog output circuit.	One or two channels may be configured.

C.8 Rate Generator Error Codes

Error Codes	Description	Action
43 (RG_CHANNEL_ERR)	Rate generator channel number is invalid.	Channels supported are AI_RG (0) and AO_RG (1).
44 (RG_COUNT_ERR)	An invalid frequency was specified for the rate generator channel.	Refer to the hardware manual for information on minimum and maximum rates supported.
45 (RG_NOT_CONFIGURED)	The rate generator was enabled prior to its configuration.	Call uRGConfigure first.

C.9 Trigger Error Codes

Error Codes	Description	Action
46 (TRIG_MODE_ERR)	The specified analog trigger mode is invalid.	Modes supported are TRIG_LOtoHI and TRIG_HItoLO.

C.10 High-speed Analog Input Error Codes

Error Codes	Description	Action
47 (AIHS_RUNNING_ERR)	An attempt was made to configure a high-speed input process while one was already running on the UDAS unit.	Call uAIHSEnable to disable the previous acquisition.
48 (AIHS_LIST_ERR)	An invalid channel sequence was specified.	The channels in the list must range from channel 0 to N (where the maximum value for N depends on the input configuration, single-ended or differential).
49 (AIHS_PARAM_ERR)	The analog input configuration list is formatted incorrectly.	The gain and range must be the same for all channels.
50 (AIHS_ZCHAN_ERR)	Auto-zero function is not permitted for high-speed analog input.	Set the autozero parameter to 0.
51 (AIHS_START_ERR)	An invalid start condition for analog input acquisition was specified.	Start conditions can be START_COMMAND or START_TRIGGER.
52 (AIHS_STOP_ERR)	An invalid stop condition for analog input acquisition was specified.	Stop conditions can be STOP_COMMAND, STOP_TRIGGER, or STOP_COUNT.
53 (AIHS_PACER_ERR)	An invalid pacer source was specified for the high-speed process.	Pacer sources are AI_PACER_EXT and AI_PACER_RGEN.
54 (AIHS_TRIGGER_ERR)	An invalid trigger source was specified for the high-speed process.	Trigger sources are AI_TRIG_HI_EXT, AI_TRIG_LO_EXT, or AI_TRIGGER_AI.

Error Codes	Description	Action
55 (AIHS_NOT_CONFIGURED)	An attempt was made to start a high-speed analog input acquisition, but no process has been configured.	Call uAIHSConfigureList first.
56 (AIHS_NOT_ENABLED)	An attempt was made to read data from a high-speed analog input acquisition process, but the process has not been started.	Call uAIHSEnable first.
57 (TRIG_NOT_CONFIGURED)	An attempt was made to enable a high-speed analog input process that depends on a trigger, but the trigger source has not been configured yet.	Call uTRIGConfigure first.
58 (TRIG_CHAN_ERR)	The analog trigger channel selected for the high-speed analog input process is not among the channels being read by the process.	The trigger channel must be one of those being acquired by the process.
59 (PAC_NOT_CONFIGURED)	The pacer signal for a high-speed analog input process has not been configured.	Call uRGConfigure first.
60 (AIHS_CLUSTER_ERR)	The cluster value passed to the high-speed analog input read call is invalid.	The uBufInfoType structure returned by the high-speed calls indicates the highest cluster number that can be read.

C.11 High-speed Analog Output Error Codes

Error Codes	Description	Action
61 (AOHS_PACER_ERR)	The pacer type specified for the high-speed analog output process is invalid.	Pacer sources are AO_PACER_EXT and AO_PACER_RGEN.
62 (AOHS_SYNCH_ERR)	An invalid trigger source for analog output was selected.	Trigger sources are AO_SYNCH_AI and AO_SYNCH_NONE.
63 (AOHS_LIST_ERR)	The channels specified for the high-speed analog output process are given in an incorrect order or an invalid channel number was specified.	When both channels are to be output, channel 0 and channel 1 must be specified, in that order, in the list.
64 (AOHS_DATA_ERR)	An attempt was made to start the high-speed analog output process without sending initial data to the UDAS.	Include a data buffer in the call to uAOHSEnable.
65 (AOHS_INUSE_ERR)	An attempt was made to enable analog output while the output process was in progress.	Call uAOHSEnable to disable the previous process.
66 (AOHS_WRITE_ERR)	An attempt was made to write new data to the analog output process, but the process is configured in repeat mode, requiring that the same data be output continuously.	To change the data in repeat mode, stop the output process and reconfigure.
67 (AOHS_RECV_ERR)	The next data buffer for high-speed analog output was not received quickly enough to maintain continuous output.	Data for continuous output must be supplied promptly. If necessary, reduce the pacer rate.
68 (AOHS_NOT_RUNNING_ERR)	An attempt was made to send new data for a high-speed analog output process, but the process was not started, or has stopped.	Process stops if AOHS_RECV_ERR occurs. Call uAOHSEnable to start the process and load the first data buffer.
69 (AOHS_CONF_ERR)	An attempt was made to start high-speed analog output, but the process has not been configured.	Call uAOHSConfigureList first.

C.12 Counter Error Codes

Error Codes	Description	Action
70 (CTR_INVALID_MODE)	The counter mode specified in the counter configuration is invalid.	Valid modes are: CTR_MODE_FREQ CTR_MODE_EVENT_LO CTR_MODE_EVENT_HI CTR_MODE_PULSE_LO CTR_MODE_PULSE_HI CTR_MODE_PERIOD
71 (CTR_NOT_CONFIGURED)	A read or measure operation was attempted on a counter channel that has not yet been configured.	Call uCTRConfigure first.
72 (CTR_READ_PROG_ERR)	A counter measurement was still in progress when an attempt was made to disable the counter.	Obtain the entire counter reading before disabling it to avoid losing data.
73 (CTR_ENABLE_ERR)	An attempt was made to reconfigure a counter channel that was running at the time.	Disable the counter before reconfiguring it.
74 (CTR_GATE_ERR)	The gate time specified when configuring a counter channel for frequency measurement mode was invalid.	Gate values must be between 1 and 65535. Recommended values are provided in USBDEFS.H.
75 (CTR_LATCH_ERR)	Attempts to read the counter channel failed. This may be caused by attempting to use an input signal with too high a frequency.	See the appropriate section in the manual for counter limitations.
76 (CTR_CHANNEL_ERR)	An invalid counter channel number was specified.	Channel number must be 0.

C.13 Thermocouple Error Codes

Error Codes	Description	Action
77 (TC_BUFFER_ERR)	Unable to allocate memory for thermocouple tables.	Reduce the memory used by other applications or processes.
78 (TC_MEMORY_ERR)	Internal memory error in thermocouple table processing.	Contact the factory.
79 (TC_FILE_OPEN_ERR)	The attempt to open a thermocouple table file failed.	Verify that the table is present in the same directory with the application, and the file has not been corrupted.
80 (TC_TABLE_READ_ERR)	The thermocouple file could not be read.	The file may be in the wrong format because it has been corrupted or because it is not a U-Link.TC file.
81 (TC_NO_DATA_ERR)	Could not locate the thermocouple files.	Ensure the U-Link TC files are installed in the same directory as the U-Link DLL.
82 (TC_NO_TABLE_ERR)	The indicated thermocouple type is not supported by this version of the driver, or the thermocouple file corresponding to the thermocouple type specified is missing.	Ensure that you are using a supported thermocouple type and that all of the .TC files are in the correct directory.
83 (TC_5B_CJC_ERR)	When using 5B37 thermocouple blocks, no CJC should be used, as the cold-junction compensation is done internally to each block.	Specify the CJC temperature correction as 0.0.
84 (TC_CJC_RANGE_ERR)	The cold-junction temperature value is out of range.	Verify that the termination panel is configured correctly for thermocouple measurements.
85 (TC_VOLTAGE_RANGE_ERR)	The thermocouple temperature is out of range.	The thermocouple may be connected improperly or may be broken.

C.14 Miscellaneous Error Codes

Error Codes	Description	Action
86 (OPTION_ERR)	An invalid option value was specified to the uSetOption call.	Options are OPTION_RG_OUTPUT and OPTION_AO_DISABLE.
87 (NO_AO_ERR)	An analog output operation was attempted on a unit which does not have analog output support.	If your unit supports analog output, contact the factory.
88 (IO_NOT_READY_ERR)	An attempt to communicate to the UDAS device occurred before the unit was fully operational after a computer reboot or after being plugged into the USB.	(internal error)
89 (AI_RESET_ERR)	An internal error occurred on the UDAS unit during a reset operation.	Contact the factory.
90 (EP3_DATA_ERR)	Internal error in USB communications with the UDAS unit.	Contact the factory.
91 (OVERLOAD_ERR)	Interrupts from digital input bits configured for change-of-state modes are occurring too rapidly. The maximum aggregate input rate cannot exceed 2kHz for more than 20ms.	Check the source of input pulses to the digital input bits. It may be necessary to debounce the output of a switch, if the output rings for a long time when changing states.
92 (BURNIN_ERR)	Quality control error.	Disconnect the unit and contact the factory.
93 (AIHS_DELAY_ERR)	Illegal value for the delay parameter of uAIHSEnable.	In Stop On Trigger mode, the delay must correspond to not more than 32767 samples.

Index

A

A/D
  conversions 2-24
  start convert rate 2-42
AI-MAX 2-3
  analog expansion box 2-18
analog input
  circuit, maximum sampling rate 2-42
  function calls 2-18
  functions A-10
  high-speed functions A-16
analog output
  channel 1 2-39
  converting voltage to count data 2-33
  disabling 2-32
  enabling 2-32
  function calls 2-31
  functions A-23
  high-speed channel list 2-32
  high-speed functions 2-32, A-25
  signals 2-41
  voltage range 2-39
analog trigger 2-24
API functions, accessing 2-1
application
  programming interface 1-2
  programming interface functions 2-1

B

bipolar
  counts format 2-26
  input range 2-19
  voltage ranges 2-42
bit
  list 2-9
  mask 2-6, 2-7, 2-8
  number 2-8
buffer
  host 2-27
  local 2-27
  one cluster of returned data 2-27

C

channel
  list
    analog output 2-31
    set up for high-speed acquisitions 2-23
  samples
    program control 2-22
    trigger event 2-22
  trigger 2-39
channels
  rate generator 2-41
circular buffer 2-25, 2-26
CJC
  converting counts values to temperature values 2-41
clock pulses 2-36
clustersize 2-27
  parameter, analog input, high-speed 2-23
compilers 1-3, B-1
compiling
  linking B-1
  U-Link programs
    Borland C++ B-1
    Microsoft C++ B-1
    Microsoft Visual Basic B-1
configuration
  uRGEnable call 2-36
configuration requirements 1-3
configuring
  analog input channel trigger 2-39
  analog trigger for mode 2-39
  analog trigger for threshold 2-39
  counter 2-14
  digital I/O bits from bit list 2-9
  digital input bit modes 2-8
constants
  definitions of 1-2
conversion and linearization of data 2-42
converting
  counts data to floating point voltage 2-19
  counts value to floating point voltage value 2-42
  floating point voltage to D/A counts 2-31
  thermocouple and CJC counts to temperture values 2-41
  voltage to count data, analog output 2-33
count

parameter
analog output 2-31
analog output, high-speed 2-9, 2-32
value 2-10
value, computing, programming rate
generator 2-42
counter
event 2-14
functions A-29
hardware 2-14, 2-36
incrementing signal edges 2-7
modes 2-14
counts
data, converting to floating point voltage 2-19
to floating point voltage 2-26
creating
digital oscilloscope application 2-39
programs 2-2
crystal-based oscillator 2-23

D

data
acquisition using a trigger 2-39
parameter 2-9
value 2-10
definitions
clustersize 2-27
of constants, file location 2-1
uAIConfigureList 2-18
uAIHSConfigureList 2-22
uAIHSEnable 2-22
uAIHSRead 2-22
uAIHSReturn 2-22
uAIHSStatus 2-22
uAIMAXEnable 2-18
uAIRead 2-18
uAOConfigureList 2-31
uAOHSConfigureList 2-32
uAOHSEnable 2-32
uAOHSStatus 2-32
uAOHSWrite 2-32
uAOWrite 2-31
uCTRConfigure 2-14
uCTREnable 2-14
uCTRRead 2-14
uCTRStatus 2-14
uDeviceClose 2-2
uDeviceOpen 2-2
uDIOBITConfigureList 2-6
uDIOBITRead 2-6
uDIOBITWrite 2-6
uDIOConfigureList 2-3
uDIORead 2-3
uDIOWrite 2-3
uRGConfigure 2-36
uRGEnable 2-36
uSetOption 2-40
uSYSInquire 2-2
uTRIGConfigure 2-39
de-initializing U-Link softaware 2-3
digital
bits, configuring as trigger inputs 2-6
I/O bit functions A-36
I/O bits, configuring 2-6
I/O port functions 2-3, A-34
input
bit modes, configuring 2-8
bits
enabling and disabling 2-8
programming normal mode 2-9
data 2-7
signal, using as counter or trigger 2-6
oscilloscope application, analog trigger 2-39
output bit
data, writing 2-8
modes, programming 2-12
disabling
& enabling digital input bits, trigger inputs 2-8
analog output 2-32
analog output signals 2-41
trigger bit 2-8
Dynamic Link Library (DLL) 2-1

E

enabling
& disabling digital input bits, trigger inputs 2-8
analog output process 2-32
analog output signals 2-41
rate generator 2-37
error code information C-1

error codes A-6, C-1
  analog input C-5
  analog output C-6
  count C-10
  digital I/O C-4
  high-speed analog input C-7
  high-speed analog output C-9
  miscellaneous C-12
  multi-function C-4
  rate generator C-6
  returned by UDAS driver DLL C-1, C-3
  thermocouple C-11
  trigger C-7
event counter 2-14
example
  programming
    analog trigger circuit with U-Link functions 2-40
    normal mode digital input bits 2-9
    normal mode digital output bits 2-12
    rate generator 2-37
    trigger mode digital input bits 2-9
  reading and writing port data 2-4
  two-dimensional array 2-19
executing U-Link functions B-2

F

floating point
  voltage value, convert counts value 2-42
  voltage, converting to D/A counts 2-31
flow chart
  digital input bit mode 2-5, 2-10
  digital output bit mode 2-13
  digital trigger input mode 2-11
  event counter block diagram 2-15
  frequency measurements 2-17
  high-speed analog input acquisition 2-29
  period measurements block diagram 2-16
  rate generator programming 2-38
  reading and writing digital I/O data 2-5
  software controlled analog input read 2-21
  writing high-speed analog output data 2-35
frequency measurement 2-16
function
  call specifications A-4
  calls A-2
  calls included in software libraries 1-1
  calls, library 1-2
functions
  initialization and deinitialization 2-2
  miscellaneous 2-40

G

gain
  and input ranges, CJC & thrermocouple readings 2-42
  configuring list of analog channels 2-18
gate
  input 2-14
  value 2-17
  values, frequency measurement 2-17

H

hardware
  counter, UDAS 2-14, 2-36
  paced high-speed analog input acquisition 2-18
header files 1-2, 2-1
high pulse 2-16
high-speed
  acquisition, sampling 2-18
  data, reading 2-26
  start and stop modes 2-24
host buffer 2-27

I

I/O
  option function A-44
import library
  API functions 2-1
  Borland C++ 2-1
  Microsoft Visual C++ 2-1
incrementing a counter 2-7
initialization
  and de-initialization functions 2-2
  function calls A-4
  functions A-7
input
  signal, counting one period 2-15

signal, counting pulses 2-14
voltage, input channel, trigger signal 2-39
installation
software 1-3
integer error code, U-Link library functions A-1
interface
programming application 2-1
internal rate generator 2-22

L
libraries
import A-4
supported u-link software A-4
library
dynamic link A-4
function calls 1-2
license agreement 1-3
list parameter 2-3, 2-32
analog output 2-31
low pulse 2-16

M
mandatory function
uDeviceClose 2-3
uDeviceOpen 2-2
maximum sampling rate, analog input & waveform output 2-42
measurement
frequency 2-16
period 2-15
pulse 2-16
resolution 2-15
measuring
duration of complete period, input signal 2-15
Microsoft Windows environment programming language support 1-3
modes
counter 2-14
stop on command 2-25
multiple
threads, drivers 2-1
transitions, gate input 2-14

O
operating system 1-3
for UDAS devices B-1
optional function
uSYSInquire 2-2
output
frequency formula 2-36
frequency range 2-36

P
pacer
high-speed analog input 2-22
pacers and triggers 2-23
start and stop modes 2-24
pacing signal 2-23
panel numbers
AI-MAX, specifying 2-18
parameter
trigger 2-24
period measurement 2-15
personal computer system 1-3
port
specifying 2-4
port bit mask 2-9
portbit parameter 2-6, 2-8
program
license agreement 1-3
source files, sample A-4
programming
application interface 1-2
digtial output bit modes 2-12
initialization sequence 2-2
interface 2-1, A-1
normal mode digital input bits 2-9
procedures 2-2
procedures, digital input and output 2-8
writing in C, declaration for array 2-19
programs
locating sample 2-2
writing, software libraries 1-3
pulse measurement 2-16
pulses
counting on input signal 2-14

R
rate generator
  channels 2-41
  error codes C-6
  function calls 2-36
  functions A-41
  output 2-41
  output frequency 2-36
reading
  analog input data 2-27
  data
    bits configured for input mode 2-9
  digital input bit data 2-7
  high-speed data 2-26
  multiple samples 2-18
reading and writing digital data 2-4
reconfiguration
  uRGEnable call 2-36
repeat flag 2-33
reset parameter 2-10
run-time library
  header files for C++ A-5
  visual basic A-5
run-time library reference A-1

S
sample programs 2-2
signal
  analog output 2-41
  conditioning module 2-42
  edges, incrementing 2-7
  externally supplied 2-22
  TTL level 2-24
single-threaded capability, U-Link drivers 2-1
software
  controlled analog input acquisition 2-18
  installation 1-3
  libraries 1-1, A-4
  libraries, introduction 1-2
  license agreement 1-3
sources
  pacer and trigger 2-24
specifying
  port list 2-3
starting data acquisition, trigger 2-24
status information
  analog input, high-speed 2-22
supported compilers 1-3
sytem configuration requirements 1-3

T
thermocouple
  error codes C-11
  measurement requirements 2-42
  table file B-2
threshold
  converting floating point voltage to counts 2-39
  value 2-39
  voltage 2-39
timebase 2-23
timing analog input samples 2-22
transitions
  multiple, gate input 2-14
trigger
  bit, disabling 2-8
  channel 2-39
  function A-45
  function calls 2-39
  high-to-low 2-9
  input bit state 2-10
  inputs 2-6
  lo-to-high 2-9
  parameter 2-24
  signal 2-39
triggers 2-24
  and pacers 2-23
TTL level signal 2-24

U
uAIConfigureList A-11
uAIHSConfigureList A-16
uAIHSEnable A-18
uAIHSRead A-20
uAIHSReturn A-21
uAIHSStatus A-22
uAIMAXEnable 2-3, A-10
uAIRead A-13
uAOConfigureList A-23

uAOHSConfigureList A-25
uAOHSEnable A-26
uAOHSStatus A-27
uAOHSWrite A-28
uAOWrite A-24
uCountsToTemperature 2-41, A-46
uCountsToVolts 2-42, A-48
uCTRConfigure A-29
uCTREnable A-29, A-31
uCTRRead A-32
uCTRStatus A-33
UDAS
  device, initialization sequence 2-2
  hardware, operating system B-1
  unit, serial and verison numbers 2-2
UDAS device
  serial number 2-2
uDeviceClose 2-3, A-8
uDeviceOpen 2-2, A-7
uDIOBITConfigureList 2-6, A-36
uDIOBITRead 2-7, A-39
uDIOBITWrite 2-6, 2-8, A-40
uDIOConfigureList 2-4, A-34
uDIOListType 2-3
uDIORead 2-4, A-35
uDIOWrite 2-4, A-36
uFrequencyToRGCounts 2-42, A-48
U-Link programs
  compiling
    Microsoft C++ B-1
  compiling Microsoft Visual Basic B-1
  compiling, Borland C++ B-1
U-Link software B-1
  compilers supported B-1
  installation 1-3
  libraries 1-3
    introduction and installation 1-1
unipolar
  counts format 2-26
  data 2-19
  input range 2-19
  voltage ranges 2-42
unipolar voltage ranges 2-42
uRGConfigure 2-42, A-41
uRGEnable A-42
uSETOption A-44
uSetOption function calls
  analog output signals 2-40
  rate generator output 2-40
uSYSInquire 2-2, A-8
utility functions A-46
uTRIGConfigure 2-39, A-45
uVoltsToCounts A-49

W

waveforms 2-32
writing digital output bit data 2-8

Z

zero-reference field 2-23